(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 October 2001 (25.10.2001)

PCT

(10) International Publication Number
WO 01/79949 A2

(51) International Patent Classification⁷: G05B 23/00

(21) International Application Number: PCT/US01/11417

(22) International Filing Date: 6 April 2001 (06.04.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/550,549 14 April 2000 (14.04.2000) US

(71) Applicant *(for all designated States except US)*: **SYMYX TECHNOLOGIES, INC.** [US/US]; 3100 Central Expressway, Santa Clara, CA 95051 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **CREVIER, Thomas** [US/US]; 1717 Cedar Lake Court, San Jose, CA 95131 (US). **GORDEYCHEV, Dmitry, O.** [RU/US]; 19109 Santa Maria Avenue, Castro Valley, CA 94546 (US). **DORSETT, David, R., Jr.** [US/US]; 3821 Marilyn Court, Pleasanton, CA 94588 (US).

(74) Agents: **PORTER, Timothy, A.** et al.; Fish & Richardson P.C., 2200 Sand Hill Road #100, Menlo Park, CA 94025 (US).

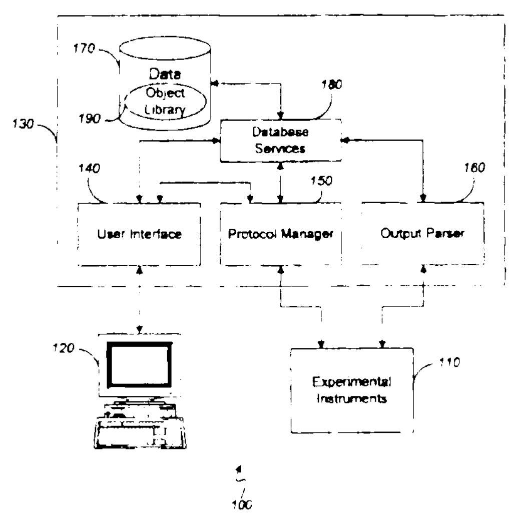
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
- *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) **Title:** AUTOMATED PROCESS CONTROL AND DATA MANAGEMENT SYSTEM AND METHODS



(57) **Abstract:** The invention provides methods, systems and apparatus, including computer program apparatus, implementing techniques for processing experimental data derived from a combinatorial library of materials. The techniques include providing an experiment protocol defining a program of actions to be carried out during a set of experiments and executing the program of actions. The protocol includes at least a first action linked to a library description of a combinatorial library. The library description includes data identifying a plurality of library positions corresponding to a plurality of members of the combinatorial library. Execution of the first action includes performing a method specified by the first action using the library description as an input.

WO 01/79949 A2

AUTOMATED PROCESS CONTROL AND DATA MANAGEMENT SYSTEM AND METHODS

TECHNICAL FIELD

This invention relates to systems for collecting and processing experimental data.

BACKGROUND

The discovery of new materials with novel chemical and physical properties often leads to the development of new and useful technologies. Traditionally, the discovery and development of materials has predominantly been a trial and error process carried out by scientists who generate data one experiment at a time. This process suffers from low success rates, long time lines, and high costs, particularly as the desired materials increase in complexity. There is currently a tremendous amount of activity directed towards the discovery and optimization of materials, such as superconductors, zeolites, magnetic materials, phosphors, catalysts, thermoelectric materials, high and low dielectric materials and the like. Unfortunately, even though the chemistry of extended solids has been extensively explored, few general principles have emerged that allow one to predict with certainty the composition, structure and reaction pathways for the synthesis of such solid state compounds.

As a result, the discovery of new materials depends largely on the ability to synthesize and analyze large numbers of new compounds. Given approximately 100 elements in the periodic table that can be used to make compositions consisting of two or more elements, an incredibly large number of possible new compounds remain largely unexplored. One approach to the preparation and analysis of such large numbers of compounds has been the application of combinatorial chemistry.

In general, combinatorial chemistry refers to the approach of creating vast numbers of compounds by reacting a set of starting chemicals in all possible combinations. Since its introduction into the pharmaceutical industry in the late 80's, it has dramatically sped up the drug discovery process and is now becoming a standard practice in the industry (Chem. Eng. News Feb. 12, 1996). More recently, combinatorial techniques have been successfully applied to the synthesis of inorganic materials (G. Briceno et al., SCIENCE 270, 273-275, 1995 and X. D. Xiang et al., SCIENCE 268, 1738-1740, 1995). By use of various surface deposition techniques, masking strategies, and processing conditions, it is now possible to generate hundreds to thousands of

materials of distinct compositions per square inch. These materials include high T_c superconductors, magnetoresistors, and phosphors.

Using these techniques, it is now possible to create large libraries of chemically diverse compounds or materials, including biomaterials, organics, inorganics, intermetallics, metal alloys, and ceramics, using a variety of sputtering, ablation, evaporation, and liquid dispensing systems as disclosed in U.S. Patents No. 5,959,297, 6,004,617 and 6,030,917.

The generation of large numbers of new materials presents a significant challenge for conventional analytical techniques. By applying parallel or rapid serial screening techniques to these libraries of materials, however, combinatorial chemistry accelerates the speed of research, facilitates breakthroughs, and expands the amount of information available to researchers. Furthermore, the ability to observe the relationships between hundreds or thousands of materials in a short period of time enables scientists to make well-informed decisions in the discovery process and to find unexpected trends. High throughput screening techniques have been developed to facilitate this discovery process, as disclosed, for example, in U.S. Patents No. 5,959,297, 6,030,917 and 6,034,775.

The vast quantities of data generated through the application of combinatorial and high throughput screening techniques can easily overwhelm conventional data acquisition, processing and management systems. Existing laboratory management systems are ill-equipped to handle the large numbers of experiments required in combinatorial applications, and are not designed to rapidly acquire and process the large amount of data generated by such experiments, imposing significant limitations on throughput, both experimental and data processing, that stands in the way of the promised benefits of combinatorial techniques.

SUMMARY

The invention provides an automated system, computer programs and methods for controlling the execution of experiments and the collection and processing of experimental data using high throughput screening technologies.

In general, in one aspect, the invention provides a process control and data management system for processing a set of experiments on a combinatorial library of materials. The system includes a memory storing one or more experiment protocols and a protocol manager operable to receive and execute an experiment protocol from the memory. Each experiment protocol defines a program of actions to be carried out during

a set of experiments. The program of actions includes a first action linking the experiment protocol to a library description stored in the memory. The library description is data including data identifying a plurality of library positions corresponding to a plurality of members of the combinatorial library. Upon execution of the first action, the protocol manager is operable to perform a method specified by the first action using the library description as an input.

Particular advantageous implementations can include one or more of the following features. The first action can be operable to cause the protocol manager to perform a specified method upon a plurality of the library positions identified by the library description. The first action can be operable to cause the protocol manager to perform a specified method sequentially upon a plurality of the library positions identified by the library description. The protocol manager can be operable to receive an input identifying a combinatorial library member as invalid and modify the library description to remove the corresponding library position from the library description. The experiment protocol can include a library creator action operable to cause the protocol manager to generate a library description corresponding to a specified library. The experiment protocol can include a library loader action operable to cause the protocol manager to retrieve a library description from the memory. The experiment protocol can include a library manipulator action operable to generate at least one output library description from at least one input library description. The library manipulator action can be operable to cause the protocol manager to retrieve the input library description from the memory; display a graphical representation of the input library description on a user interface; and receive an input from a user defining one or more output library descriptions based on the displayed representation. The library manipulator action can be operable to cause the protocol manager to retrieve the input library description from the memory and apply a selection criterion to define one or more output library descriptions based on the input library description. The protocol manager can be operable to generate during execution of an experiment protocol a stream of process data including output data received from one or more experimental instruments coupled to the protocol manager. The system can include an output parser operable to retrieve the stream of process data and store the process data in the memory. Each experiment protocol can include a plurality of blocks, each including at least one action operable upon execution by the protocol manager to receive input data and generate output data, as well as instructions for sequentially passing input data to and receiving output data from the actions in the

block. The protocol manager can be operable to execute a first block in the protocol, such that as each action in the first block is executed the protocol manager adds the corresponding output data to the process data stream. The memory can store process control objects including one or more display objects operable to cause the protocol manager to display the output data generated by an action in a protocol. The protocol manager can be operable to execute a display object such that the output data from an action in an executing protocol is displayed on an output device in a format specified by the display object. The protocol manager can be operable to execute upon completion of execution of the first block a second block identified from the plurality of blocks in the protocol by a flow control instruction generated upon completion of execution of the first block. The protocol manager can be operable to generate execution status data upon execution of each action and each block and add the execution status data to the process data stream, the execution status data designating an execution status for the action or block. The protocol manager can be operable upon execution of the first block to use the execution status data for the block as the flow control instruction identifying the second block to be executed from the plurality of blocks in the protocol. The memory can include a library of actions, each action being defined by action methods and action data, The system can include a user interface module operable to cause the protocol manager to create one or more blocks and to include in each block one or more actions selected from the library of actions and to define a flow of control among the one or more blocks. The library of actions can include one or more detector objects operable upon execution to retrieve experimental data from a detector and to provide the experimental data as output data to the protocol manager. The library of actions can include one or more result objects operable upon execution to receive experimental data from a detector object, calculate one or more properties from the experimental data, and provide the calculated properties as output data to the protocol manager. The protocol manager can be operable to execute an experiment protocol to carry out an experiment on a combinatorial library of materials, the protocol including a block operable to test the output data stream, define a new library description that corresponds to a subset of the combinatorial library of materials, and to cause at least a subset of the blocks of the experiment protocol to operate on the subset of the combinatorial library of materials corresponding to the new library description.

In general, in another aspect, the invention provides methods and apparatus, including computer program apparatus, implementing techniques for processing a set of

experiments on a combinatorial library of materials. The techniques can include providing an experiment protocol defining a program of actions to be carried out during a set of experiments and

executing the program of actions. The protocol includes at least a first action linked to a library description of a combinatorial library. The library description is data including data identifying a plurality of library positions corresponding to a plurality of members of the combinatorial library. Execution of the first action comprises performing a method specified by the first action using the library description as an input.

Particular advantageous implementations can include one or more of the following features. The first action can be linked to the library description at run time. The specified method can be performed upon all of the library positions identified by the library description. The specified method can be performed sequentially upon each of the library positions identified by the library description. The method can include receiving an input identifying a combinatorial library member as invalid; and modifying the library description to remove the corresponding library position from the library description. The experiment protocol can include a library creator action operable to cause the protocol manager to generate a library description corresponding to a specified library and store the library description in a memory. The experiment protocol can include a library loader action operable to retrieve a library description from a memory. The experiment protocol can include a library manipulator action operable to generate at least one output library description from at least one input library description. The method can include, upon execution of the library manipulator action, retrieving the input library description from a memory; displaying a graphical representation of the input library description on a user interface; and receiving an input from a user defining one or more output library descriptions based on the displayed representation. The method can include, upon execution of the library manipulator action, retrieving the input library description from a memory and applying a selection criterion to define one or more output library descriptions based on the input library description. The protocol can include a plurality of blocks, each block specifying one or more actions and instructions for sequentially passing input data to and receiving output data from each action in the block. Executing the program of actions can include, as each action in a first block is executed, adding the corresponding output data to a process data stream; and, upon completion of execution of the first block, executing a second block in the protocol according to a flow control instruction determined upon completion of execution of the

first block. The method can include asynchronously retrieving the output data from the process data stream and storing the output data in a memory. The method can include providing a display object operable upon execution to display the output data associated with an action in the protocol; and displaying the output data on an output device in a format specified by the display object. The method can include, upon execution of each block in the protocol and each action in each block, generating execution status data identifying a completion status for the corresponding block or action; adding the execution status data to the process data stream; and asynchronously retrieving the execution status data from the process data stream. The method can include providing a library of pre-defined actions; providing the experiment protocol can include receiving user input defining an arrangement of blocks in the protocol and a selection and arrangement of actions in each block from the library of actions.

In general, in another aspect, the invention provides methods and apparatus, including computer program apparatus, implementing techniques for processing a set of experimental data derived from a combinatorial library of materials. The techniques can include providing an experiment protocol including a program of actions for processing a set of experimental data, retrieving a set of experimental data derived from the combinatorial library in response to execution of a first action, and performing the specified data processing method on the set of experimental data in response to execution of a second action. The experiment protocol includes at least a first action linked to a library description of a combinatorial library, and a second action specifying a data processing method. The library description is data including data identifying a plurality of library positions corresponding to a plurality of members of the combinatorial library.

Particular advantageous implementations can include one or more of the following features. The set of experimental data can be retrieved from an experimental instrument. The set of experimental data can be retrieved from a memory.

Advantages that can be seen in implementations of the invention include one or more of the following. The implementation of process functionality as component objects makes the system readily extensible to encompass new synthetic and analytical techniques and apparatus without requiring wholesale modification to the overall system. Isolation of data collection and processing steps from the experimental process can help to protect data against equipment failure and ensures that all data from an experiment is stored for later retrieval and analysis. Implicit database linkages permit users to search

and manipulate all data from an experiment without requiring users to have explicit knowledge of the database format.

The details of one or more embodiments of the invention are set forth in the accompanying drawings and the description below. Other features, objects, and advantages of the invention will be apparent from the description and drawings, and from the claims.

DESCRIPTION OF DRAWINGS

FIG. 1 is a is a block diagram of an automated laboratory management system including a process control and data management system according to the invention.

FIG. 2 is a flow diagram illustrating a protocol for controlling an automated laboratory management system.

FIG. 3A is a flow diagram illustrating a process for generating an experiment procedure.

FIG. 3B is a window illustrating a user interface for generating an experiment procedure implemented on a process control and data management system.

FIG. 4A is a flow diagram illustrating a process for generating an experiment protocol for controlling an automated laboratory management system.

FIGS. 4B to 4E are windows illustrating a user interface for graphically defining a protocol.

FIG. 5 is a window illustrating a protocol for controlling a high throughput gel permeation chromatography experiment.

FIG. 6 is a flow diagram illustrating the execution of the protocol of FIG. 5 by a process control and data management system.

Like reference symbols in the various drawings indicate like elements.

DETAILED DESCRIPTION

Referring to FIG. 1, an automated laboratory management system 100 is configured to control a set of automated experiments involving a combinatorial library of materials and to manage data generated during the course of the experiments. Laboratory management system 100 includes one or more automated experimental instruments 110, one or more input/output devices 120, such as a keyboard, mouse, video monitor or the like, and a process control and data management system 130. Process control and data management system 130, in turn, includes a user interface module 140 coupled to

input/output devices 120, a protocol manager 150 coupled to automated apparatus 110 and an output parser 160, which is also coupled to apparatus 110. User interface module 140, protocol manager 150 and output parser 160 are coupled to database 170 through database services module 180. In one implementation, user interface module 140, protocol manager 150 and output parser 160 are implemented as computer programs stored on computer-readable media, and can be distributed across multiple computers. Experimental instruments 110 can include any automated device for handling or processing materials, gathering experimental data or analyzing chemical properties, such as synthesis robots, pumps, temperature controllers, reactors, digital cameras for visible or infrared imaging, evaporative light scattering detectors, microcalorimetry arrays and the like.

Examplary experimental apparatus 110 for material processing and screening are disclosed in U.S. Patent Application No. 09/941,170, filed September 30, 1997; U.S. Patent Application No. 09/156,827, filed September 18, 1998; U.S. Patent Application No. 08/841,423, filed April 22, 1997; and U.S. Patent Application No. 09/237,502, filed January 26, 1999; U.S. Patent Application No. 09/093,870, filed June 9, 1998; U.S. Patent Application No. 09/300,634, filed April 27, 1999; U.S. Patent Application No. 09/039,991, filed March 16, 1998; U.S. Patent Application No. 09/067,448, filed April 27, 1998; U.S. Patent Application No. 09/227,558, filed January 8, 1999; U.S. Patent Application No. 09/285,363, filed April 2, 1999; U.S. Patent Application No. 09/285,393, filed April 2, 1999; U.S. Patent Application No. 09/285,333, filed April 2, 1999; U.S. Patent Application No. 09/285,335, filed April 2, 1999; U.S. Patent Application No. 09/285,392, filed April 2, 1999; U.S. Patent Application No. 09/410,546, filed October 1, 1999; U.S. Patent Application No. 09/414,744, filed October 8, 1999; U.S. Patent Application No. 08/946,135, filed October 7, 1997; U.S. Patent No. 5,959,297; U.S. Patent 5,985,356; U.S. Patent No. 6,030,917; U.S. Patent No. 6,034,775; U.S. Patent Application No. 09/033,207, filed March 2, 1998; U.S. Patent Application No. 09/174,986, filed October 19, 1998; U.S. Patent Application No. 09/417,125, filed November 19, 1998; U.S. Patent Application No. 09/177,170, filed October 22, 1998; U.S. Patent Application No. 09/211,982, filed December 14, 1998; U.S. Patent Application No. 09/293,223, filed January 29, 1999; U.S. Patent Application No. 09/474,344, filed December 29, 1999; U.S. Patent Application No. 09/112,247, filed July 8, 1998; U.S. Patent Application No. 09/149,586, filed September 8, 1998; U.S. Patent Application No. 09/458,398, filed December 10, 1999; U.S. Patent Application No.

09/215,417, filed December 18, 1998; U.S. Patent Application No. 09/205,071, filed December 4, 1998; U.S. Patent Application No. 09/518,794, filed March 3, 2000; U.S. Provisional Patent Application No. 60/157,338, filed October 1, 1999; and WO 97/32208. Each of these patents and patent applications is incorporated herein by reference. As used in this specification, an experiment can include using experimental instruments to manipulate samples and/or gather data, or processing data gathered during previous experiments.

Database 170 includes a library of pre-defined objects and object prototypes 190 that process control and data management system 130 uses to control automated laboratory management system 100. Protocol and procedure prototypes in library 190 define container objects in which process authors will specify a sequence of actions to be executed by process control and data management system 130 to carry out an experiment, as will be described in more detail below. Action prototypes in library 190 define a number of actions that provide the basic functionality of process control and data management system 130. Actions are defined by a set of properties, which can include some combination of a name, one or more input parameters and one or more output parameters (which are themselves simple objects having a name or type and a value that may be a number, string, object or array of any of these), and a set of one or more action methods defining how the action operates on its input and/or output parameters.

Each action can have an associated "well count" (e.g., a set of combinatorial library members, such as wells on a library substrate) to which one or more of the action's methods are to be applied. In one implementation, an action can have an associated well count of none, meaning that the action's methods are not applied to any members of a combinatorial library, one, meaning that the action's methods are to be applied sequentially to all positions in a corresponding library, a subset, meaning that the action's methods are to be applied sequentially to groups of positions of a library, or all, meaning that the action's methods are applied to all library locations at once. Actions whose methods require sequential access to individual library locations (i.e., actions having one or subset well counts) receive the current "active" library location as an input parameter; those whose methods require access to all library locations (i.e., actions having all well counts) receive the entire library (i.e., all library locations) as an input parameter.

An action's input parameters include all data needed to execute the action's methods, and may include other objects to be called by the action's methods, such as a

library object to which the action's methods are to be applied, system resources or output parameters of previous objects in a procedure or protocol. Likewise, an action's output parameters include all data generated by execution of the action's methods, including, for example, the system resource the action connects to, or peak height, width, and area values generated by execution of a peak detector action.

Particular combinatorial libraries are represented as library objects stored in database 170 and are used by process control and data management system 130 to associate actions with specific library positions. In one implementation, a library object contains a full description of a combinatorial library, including the geometry of the physical library substrate (e.g., 8 rows by 12 columns), which positions on the substrate are valid and information about where, if at all, the substrate is located on a work surface available to experimental instruments 110.

Process control and data management system 130 can implement actions create, load and manipulate library objects. A first class of such actions includes library creators, which are actions that generate new libraries that are not yet stored in database 170. In one example, a "Create Library Action" generates a library object representing a substrate in a high throughput screening system. The Create Library Action has two input parameters: "Library ID", which permits the procedure author to specify an ID for the library object, and "User Prompted?", which permits the author to specify whether or not the Library ID value can be changed at run time. The action has two output parameters as well: "Library ID", which identifies the actual ID (i.e., the corresponding input parameter or some other value if altered by the user), and "Library Object", which contains the actual library object itself. Upon execution by protocol manager 150, the Create Library Action creates a library object of specified dimensions (e.g., a 12x12 matrix) populated by valid positions according to the particular physical substrate in question.

A library loader action loads an existing library object from database 170, and may yield additional information about the library as well. In one example, a "Load Standards to Deck Action", retrieves from database 170 a standards library, such as may be used in a gel permeation chromatography protocol as discussed below. The Load Standards to Deck Action has input parameters including "Library ID", which specifies an initial value for the ID of the library object, and "Robot System Resource", which identifies robot resource that controls the workspace onto which the physical library is to be loaded. The output parameters include "Library ID", corresponding to the library object that is actually loaded, "Library Object", containing an output library object, and

"Molecular Weights", which contains the molecular weights of the standards. When protocol manager 150 executes the action, it prompts the user for a library number, a substrate type (e.g., what sort of rack it is in), and the rack position on the deck or work surface. Protocol manager 150 loads the library from database 170 (if it exists) and populates the library object appropriately. Protocol manager 150 also reads the molecular weights of the standards out of database 170 and returns them through the action's Molecular Weights output parameter. Other library loader actions may load different kinds of libraries (e.g., sensor array module libraries for measuring thermal properties), or may return different types of information (e.g., a "Load Analyzed Library Action" may return the measured data for a library in a gel permeation chromatography or other experiment).

Library manipulator actions generate new libraries from existing ones. In one example, a "Create Daughter Library Action" takes as input parameters a "Master Library Object" parameter containing a library from which a daughter library is to be created, and a "Robot System Resource" parameter identifying a robot resource that controls the workspace on which the daughter plate is to be created. The action's output parameter is "Daughter Library Object", which contains the replicated daughter library. When protocol manager 150 executes the Create Daughter Library Action, it prompts the user for the location of the daughter plate on the work surface and replicates the specified master library object at the specified new deck location. Protocol manager 150 returns the replicated daughter library to the user as the Daughter Library Object parameter.

Another example of a library manipulator is a "User Selection Library Action." This action has an input parameter "Source Library" and two output parameters "Selected Positions Library" and "Unselected Positions Library." When protocol manager 150 executes the action, it presents the user with a graphical representation of the specified source library through user interface 140. The user can then select and de-select positions in the library through input devices 120 – for example, selecting positions exhibiting a desired visual characteristic in the library's graphical representation. When selection is complete, protocol manager 150 creates and returns two new libraries, one corresponding to the selected positions and the other to the unselected positions, which protocol manager 150 can then use in further process steps.

Still another library manipulator can be implemented as an "Auto-Selection Library Action" similar to the User Selection Library Action just discussed. Rather than permit the user to select positions, however, when protocol manager 150 executes this

action it automatically selects library positions from a source library (designated in an input parameter) by applying a specified logical expression (another input parameter). The action returns two output parameters – an "Accepted Library" corresponding to positions that passed the logical test, and an "Unaccepted Library" of positions that failed the test.

Library 190 also includes a series of system resource prototypes that define the interaction between process control and data management system 130 and particular physical devices or systems associated with automated laboratory management system 100, such as experimental instruments 110. In general, process control and data management system 130 requires that database 170 include a system resource prototype for every type of physical device or system associated with system 100. Thus, for example, to implement a protocol employing experimental instruments 110 including a heater controller, a pump and a detector, database 170 must include: a heater controller system resource prototype defining heater controller properties such as an object name for a particular instance of the controller resource, a number of channels, and a com port to which the controller is attached; a pump system resource prototype, defining pump properties such as an object name for a particular instance of the pump resource, high and low pressure limits, a head size, a pump address, and a com port to which the pump is attached; and a detector system resource prototype, defining detector properties such as an object name for a particular instance of the detector resource and a com port to which the detector is attached. In order to control the acquisition of data from the detector, process control and data management system 130 may also call an instance of a data gatherer system resource prototype, defining relevant data collection properties such as, for example, a particular data acquisition frequency. Because system resource objects provide process control and data management system 130 with all relevant information about the corresponding physical devices or systems, extension of process control and data management system 130 to operate with new or different experimental instruments 110 requires only the preparation of a new system resource object prototype corresponding to the new instrument and, if necessary, new action object prototypes to provide additional functionality to control the new instrument and process data it produces.

Finally, library 190 also includes a number of pre-defined objects that provide specific functionality for use by action objects, including, for example, peak detectors, peak integrators, numeric transforms, calibration objects, data acquisition and parsing and

the like.

Referring to FIG. 2, to control the operation of experimental instruments 110 during an experiment, protocol manager 150 retrieves an experiment protocol 200 from database 170 through database services module 180. A protocol 200 is a series of predefined experimental procedures (represented as blocks 210) that controls the execution of the experiment, including, for example, initialization of experimental instruments 110, setting of reaction conditions such as temperature and pressure, the operation of the robotics that perform the experiment, the collection of experimental data and the processing of that data to obtain experimental results. A protocol 200 follows a flowchart of linked blocks arranged in a logical order of execution. Each block, in turn, includes a series of functional steps, or actions 220, to be performed by automated laboratory management system 100 in the course of the experiment, arranged in a specific order of execution. Blocks 210 can also define specific functionality such as variables for use by other procedures in a protocol, flowchart logic – for example, switch block 230 – to provide for branching in response to measurement of experimental conditions or results and error handling – such as exception block 240 – to address exceptions that occur as the experiment progresses. Actions 220 include, for example, instructions directing the collection of raw data from a detector or instructions for identifying peaks in the collected data, as will be discussed in more detail below. In one implementation, a protocol 200 is a container object containing multiple block objects representing experimental procedures 210. Block objects, in turn, contain one or more action objects representing actions 220.

As will be explained in more detail with respect to FIGS. 4A through 4E below, a author creates an experiment protocol using input/output devices 120, user interface module 140 and protocol manager 150 to assemble pre-defined procedure blocks retrieved from database 170 via database services module 180 into a sequence of steps defining a desired experiment. Procedures may be obtained from memory, or they may be created by the author through input/output devices 120 and user interface module 140. Referring to FIG. 3A, the author initiates a procedure definition method 300 by providing an appropriate input. In one implementation, in response to an input selecting an appropriate sequence of entries in a hierarchy of menus displayed by user interface module 140 on output device 120, protocol manager 150 instantiates a procedure prototype from database 170 and causes user interface module 140 to display a corresponding procedure template 350 as shown in FIG. 3B. After specifying a procedure name in field 355 (step 310), the author selects an action from a list 360 of

available actions (step 320). Protocol manager 150 instantiates the corresponding action prototype from database 170 and causes user interface module 140 to display a corresponding entry in Parameters pane 365 of workspace 370.

After selecting a desired action, the author optionally specifies any default value for one or more of the action's parameters (step 330), for example, by selecting a particular parameter 375 and entering a value in Value field 380. If additional actions are needed to complete the procedure (the YES branch of step 340), steps 320 and 330 are repeated as required. When all necessary actions have been selected and any desired parameter values have been defined (the NO branch of step 340), the procedure is saved in database 170 for later use (step 345).

Referring to FIG.4A, when all necessary procedures have been created and loaded into database 170, either generated by the author or obtained from other sources, the author creates a protocol by initiating protocol definition method 400 (for example, by selecting an appropriate sequence of entries in a hierarchy of menus displayed by user interface module 140 on output device 120). In response to this input, protocol manager 150 calls a protocol prototype in database 170 and causes user interface module 140 to display a corresponding protocol template 440 on output device 120, as shown in FIG. 4B. The author names the new protocol (step 405), for example, by entering a name in field 442, and optionally provides additional identifying in fields 444 and 446. The author adds a first block to the protocol, for example, by selecting an icon representing the desired block type from a menu or toolbar 448 (step 410), causing user interface module 140 to add an appropriate block to protocol 450 in workpane 452. User interface module 140 also causes output device 120 to display an appropriate editor dialog 454 in which the author can specify the information to define the new block (step 415), as shown in FIG. 4C. Block types can include, for example, variable blocks 456 for creating and defining variables, switch blocks 458 for creating flow control logic, experiment blocks 460 for adding pre-defined procedures, and exception blocks 462 for defining exception handling rules.

Variable blocks 456 set the values of variables, which are containers for values. Alternatively, output parameters can be mapped to variables to assigned variable values. When a variable block is added to a protocol, variable editor dialog 454 prompts the author to provide a block name in field 464, and a variable name (entered by the author or selected from a list of previously defined variables) in field 466. The author can assign a value (either a numeric value or the value of another variable) in First Operand field 468.

To define a more complex variable, the author may optionally select a numeric operator button 470 (e.g., plus or minus) and specify a second operand (again, either a numeric value or a variable) in corresponding field in workspace 472. The author may position the block as desired in workpane 452, for example, by selecting the block and dragging it to a desired location using a mouse or other input device. Finally, the author can link the new block to one or more other blocks in the protocol, for example by selecting a previously defined block and dragging a connection to the new block (step 420). The author repeats steps 410 to 420 to add additional blocks to the protocol as required (the YES branch of step 425). When all desired blocks have been added (the NO branch of step 425), the completed protocol is saved in database 170 for execution by process control and data management system 130 (step 430).

Switch blocks 458 create branched flow control logic that depends on the evaluation of a specified logical expression. When a switch block is added to a protocol, switch editor dialog 474 (shown in FIG. 4D) prompts the author to provide a block name and operands as described above in connection with variable editor dialog 454. The author is also prompted to specify a comparitor using buttons 476, which defines the logical operation (e.g., greater than or equals) with which protocol manager 150 will compare the operands the author has identified. As described above, the author can move the switch block to any desired location, and can link the block to a previously defined block or blocks by selecting the previous block and dragging a connection to the new switch block. When the connection is created, user interface module 140 prompts the author to specify whether the link is to occur on success or on failure of the specified logical operation.

Experiment blocks 460 are added to protocols as containers for previously-defined procedures. After an experiment block is added to a protocol, the author adds the desired procedure by selecting the procedure from a list of available procedures and dragging the procedure to the experiment block in the flowchart workspace. Protocol manager 150 creates an instance of the desired procedure, assigning the procedures name and any default parameter values to the new block as defaults. The author may change any of these values in experiment editor dialog 480 (shown in FIG. 4E), which is displayed on output device 120. Default parameter values for actions in the procedure that are objects or variables are also defined in experiment editor dialog 480. Input parameters are defined by selecting input tab 482 and then selecting the appropriate action in parameters list 484. Selection of a parameter for the specified action activates a value or variable

field in workspace 488 (which will display any default value assigned during creation of the corresponding procedure, as described above), in which the author can assign a numeric value or variable name to the selected input parameter. Output parameters can be mapped to variables by selecting output tab 486. If the procedure corresponding to the experiment block contains one or more actions whose properties include a specified well count as described above, the author is prompted to specify a library object for the experiment block.

As described above, the author can move the experiment block to any desired location in the protocol flowchart, and can link the block to a previously defined block or blocks by selecting the previous block and dragging a connection to the new switch block. When the connection is created, user interface module 140 prompts the author to specify whether the link is to occur on success or on failure of the experiment block (i.e., if the specified procedure runs successfully or experiences an error).

In one implementation, process control and data management system 130 is used to control a series of high throughput gel permeation chromatography experiments carried out by automated laboratory management system 100 equipped with experimental instruments 110 including one or more material handling robots, pumps, heater controllers, chromatography columns and evaporative light scattering detector (ELSD) devices, as is generally described in co-pending U.S. applications 09/285,333, 09/285,335, 09/285,363, 09/285,392 and 09/285,393, all filed on April 2, 1999 and incorporated by reference herein. A set of actions defined for such an implementation, with corresponding associated well counts, input parameters and output parameters, is set out in Table 1, below.

Table 1: Summary of Actions for GPC Experiments

Name	Well Affinity	Input Parameters	Output Parameters
Abort Robot Program	none	Robot Program to Abort	one
		Robot System Resource	
Auto Zero ELSD Detector	none	ELSD Detector System Resource	none
Calculate Properties	one	Calibration Object	Mw
		Peak Detector Type	Mn
		Signal Threshold	Peak Area
		Derivative Threshold	

		Begin Time	
		End Time	
		Transform Type	
		Peak Integration Type	
Calibrate	One, subset or all	Curve Type	Calibration Curve Object
		Transform Type	
		Molecular Weights	
		Retention Times	
Collect Data with Trigger	one	Data Gatherer System Resource	XY Data Object
		Trigger Voltage	
		Trigger Timeout	
		Post-Trigger Delay	
		Total Acquisition Time	
Connect to Data Gatherer	none	Data Gatherer Name	Data Gatherer System Resource
		Acquisition Frequency	
Connect to ELSD Detector	none	ELSD Detector Name	ELSD Detector System Resource
Connect to Pump	none	Pump Name	Pump System Resource
Connect to Heater Controller	none	Heater Controller Name	Heater Controller System Resource
Connect to Robot	none	Robot Name	Robot System Resource
Disconnect from Data Gatherer	none	Data Gatherer System Resource	none
Disconnect from ELSD Detector	none	ELSD Detector System Resource	none
Disconnect from Pump	none	Pump System Resource	none
Disconnect from Heater Controller	none	Heater Controller System Resource	none
Disconnect from Robot	none	Robot System Resource	none
Detect Single Peak	one	Signal Threshold	Peak Height

		Derivative Threshold	Peak Area
		Begin Time	Peak Width
		End Time	Peak Width at Half Max
		XY Data Object	
Initialize Data Gatherer	none	Data Gatherer System Resource	none
Initialize Robot	none	Initialization Program Name	none
		Robot System Resource	
Inject	one	Injection Program Name	none
		Injection Volume	
		Injection Port	
		Robot System Resource	
Run Robot Program	none	Robot Program Name	none
		Robot Program Variable Names	
		Robot Program Variable Values	
		Robot System Resource	
Set ELSD Mode	none	ELSD Mode	none
		ELSD Detector System Resource	
Set ELSD Properties	none	Nebulizer Temperature	none
		Evaporator Temperature	
		Gas Flow	
		ELSD Detector System Resource	
Set Heater Temperature	none	Channel Temperature	none
		Heater Controller System Resource	
Set Heater	none	Temperatures	none

Temperatures		Heater Controller System Resource	
Start Pump	none	Flow Rate	none
		Flow Rate Ramp	
		Pump System Resource	
Wait	none	Time to Wait	none
Wait for ELSD Detector	none	Max Wait Time	none
		ELSD Detector System Resource	
Wait for Heaters	none	Max Wait Time	none
		Heater Controller System Resource	
Wait for Pump	single well	Injection Program Name	none
		Injection Volume	
		Injection Port	
		Robot System Resource	

To control the GPC experiments, process control and data management system 130 executes a protocol that causes a material handling robot to inject polymer standards into a column and derives a calibration curve from the collected peak data, and then causes the robot to inject multiple polymer samples into the column, collects peak data for the samples and calculates physical properties from that data. In one implementation, process control and data management system 130 can be configured to control the material handling robot directly using material handling actions incorporated in experiment blocks as described above. Alternatively, a GPC control protocol executed by process control and data management system 130 can call pre-defined material handling programs generated using application-specific software, such as procedures generated using Impressionist™ software available from Symyx Technologies, Inc. of Santa Clara, California and described in co-pending application serial number 09/305,830, filed May 5, 1999, which is hereby incorporated by reference for all purposes. The following example assumes that process control and data management system 130 has access to previously created material handling programs for controlling a automated synthesis robot

including an Initialize Devices program to initialize the material handling robot and an Inject Samples program to control the injections of standards and polymer samples into the detector port.

The author first creates and stores the procedures that will be used to assemble the GPC protocol, as described above with respect to FIGS. 3A-B. These include procedures to initialize data gatherer, robot, heater and detector resources, load standard and sample libraries, collect data for calibration of the evaporative light scattering detector using polymer standards, detect peaks resulting from injection of the polymer standards, calibrate the retention times of the polymer standards, collect data for the polymer samples and calculate the results for the polymer samples.

To create a procedure to initialize the data gatherer, the author initiates the procedure definition method, for example by selecting an appropriate button or menu item. After naming the procedure (in this case "Start Data Gatherer"), the author selects a Create DataGatherer Action from the list of available Data Gatherer actions 362 shown in Actions tree 360 in FIG. 3B, which will instantiate a Data Gatherer system resource. The author enters a value for the DataGatherer Name parameter, setting the name of the data gatherer object to be created, and a value for the action's Acquisition Frequency parameter, setting a rate at which data is to be collected in cycles per second. Next, the author adds an Initialize DataGatherer Action to the procedure, which will initialize the data gatherer resource. The lone input parameter for the Initialize DataGatherer Action, the DataGatherer System Resource parameter variable that will be defined as an output parameter of the Create DataGatherer Action, is not set until the Start Data Gatherer procedure is added to the protocol. The procedure is then saved in database 170 for later use.

Similarly, the author creates a procedure to initialize the robot resource, again initiating the procedure definition method and naming the procedure (for example, Start Robot) as described above. The author first adds a Create Robot Action to the procedure to instantiate a Robot system resource, and enters a value for the Robot Name parameter, identifying the system resource to be created – in this case, Local Robot. The author then adds an Initialize Robot Action to the procedure and enters a value for the action's Initialization Program Name input parameter, identifying the pre-defined Initialize Devices program to be used to initialize the material handling robot as described above. The Initialize Robot Action's Robot System Resource parameter is not defined until the Start Robot procedure is added to the protocol. The

procedure is saved in database 170 for future use. The author also generates similar procedures to initialize resources for heaters and an evaporative light scattering detector.

The author creates a Load Library procedures to load a library (e.g., of samples to calibrate the detector or of polymer samples for analysis). The author adds a Load Library Action, optionally identifies a default library object in the action's Library ID input parameter, and sets a value for the action's User Prompted? input parameter depending on whether the user is to be prompted to identify a library at runtime. The final parameter of the action, Robot System Resource (which identifies the robot resources that will have access to the respective libraries) is not set until the Load Library procedure is added to the protocol.

Next, the author creates a procedure to collect data for the standards and sample libraries (called Collect Data). The author first adds an Inject Action to the procedure, defining the following parameters: Injection Program Name, for which the author identifies the Inject Samples program for controlling the robot's injection of samples into the detector port as described above; Injection Volume, for which the author enters a value for the volume of samples to be injected; and Injection Port, the name of the injection port through which the robot is to inject. The final parameter, Robot System Resource (which identifies the robot resource to be used for the injection) is not set until the Collect Data procedure is added to the protocol. The author then adds a Collect Data With Trigger Action to collect Evaporative Light Scattering Detector (ELSD) data and defines the following parameters: Trigger Voltage, which sets the size of the voltage pulse that will trigger data acquisition; Trigger Timeout, which sets the length of time the system waits for the trigger before data collection is aborted; Post-Trigger Delay, which sets the number of seconds after the trigger occurs before data acquisition begins; Total Acquisition Time, which sets the time for which data will be collected after the trigger occurs. Again, the final parameter, the DataGatherer System Resource parameter, is not set until the Collect Data procedure is added to the protocol. The procedure is saved in database 170 for future use.

Next, the author creates a procedure to detect peaks of the injected samples (here given the name Detect Peaks). The author adds a Detect Single Peaks Action to the procedure, defining the following parameters: Peak Detector Type, for which the author selects an object from a menu or dropdown list of pre-defined objects corresponding to

the types of detectors available to automated laboratory management system 100; Signal Threshold, which sets the threshold used to detect signal above the noise; Derivative Threshold, which sets the threshold used to differentiate peaks from the background; Beginning Time, which sets the time at which data collection is to begin; and Ending Time, which sets the time at which data collection is to end. The author then saves the procedure in database 170 for future use.

Next, the author creates a procedure to calibrate the retention times of the polymer standards (here given the name Calibrate Standards). The author adds a Calibrate Action to the procedure and defines the appropriate input parameters, including the following: Curve Type, for which the author selects a pre-defined object defining the mathematical model to be used to generate a calibration curve (such as first order or third order polynomial); Transform Type, such as a logarithmic transform; Molecular Weights; and Retention Times. The author then saves the procedure in database 170 for future use.

The author then creates a procedure to calculate the results molecular weight (both weight and number averages) and peak area for the polymer samples (Calculate Results). The author adds a Calculate Properties Action to the procedure and defines the following properties: Peak Detector Type, for which the author selects a pre-defined object defining the type of peak detector to be used, such as gaussian or non-gaussian; Signal Threshold, which sets the threshold used to detect signal above the noise; Derivative Threshold; Beginning Time, which sets the time at which data acquisition was to begin; Ending Time, which sets the time at which data collection was to end; Transform Type; Peak Integration Type, for which the author selects a pre-defined object defining the type of peak integration used for the calibration. The author saves the procedure in database 170 for future use.

After these procedures have been created and stored, the author creates a protocol for the GPC experiment (here, Calibrate Standards and Calculate Sample Peaks protocol 500) as shown in FIG. 5. The author initiates the protocol creation method and names the protocol, causing user interface 140 to display a protocol template on output device 120. The author adds experiment blocks 503 and 507 that will contain the previously defined Load Library procedures to the the protocol. For each block, the author selects the previously-defined Load Library procedure from a list of stored procedures 515 and drags it into the new experiment blocks in flowchart 500, causing user interface module 140 to display experiment editor dialog 480 on output device 120. Any previously-defined default parameter values from the procedures are carried over to the new experiment

blocks; the author adds any remaining values in experiment dialog 480, for example, setting the Library ID parameter to Standards Library for block 503 and to Unknown Library for block 507. The author links the Load Standards Library and Load Samples Library blocks to the start block and Load Standards Library block, respectively, specifying that each is to be executed upon successful completion of the previous block.

Next, the author adds an experiment block 510 that will contain the previously-defined Start Data Gatherer procedure to the protocol. The author selects the Start Data Gatherer procedure from a list of stored procedures 515 and drags it into the new experiment block in flowchart 500, causing user interface module 140 to display experiment editor dialog 480 on output device 120. The previously-defined default parameter values for the Start Data Gatherer procedure are carried over to the new experiment block; the author adds the remaining object and variable parameter values in experiment dialog 480. The author defines the Data Gatherer System Resource output parameter for the procedure's Create DataGatherer Action by naming the system resource (e.g., "DataGathererSystemResource") to be created by that action. The author then defines the Data Gatherer System Resource input parameter for the procedure's Initialize DataGatherer Action, selecting the variable name DataGathererSystemResource from a dropdown list of available variables. The author then links the Start Data Gatherer block to variable block 505, specifying that the Start Data Gatherer block is to be executed upon successful execution of the variable block.

The author adds another experiment block 520 to contain the previously-defined Start Robot procedure to initialize the synthesis robot. The author selects the Start Robot procedure from a list of stored procedures and drags it into the new experiment block in flowchart 500, causing user interface module 140 to display experiment editor dialog 480 on output device 120. The author sets the remaining action parameters in the experiment editor dialog, including the Create Robot Action's Robot System Resource output parameter (set, e.g., to "RobotSystemResource"). The author then sets the Initialize Robot Action's Robot System Resource input parameter by selecting RobotSystemResource from the dropdown list of available variables. The author then links the Start Robot block to Start Data Gatherer block 510, specifying that the former block is to be executed upon successful execution of the latter.

Next, the author adds another experiment block 525 to contain the previously-defined Collect Data procedure for collecting data for the standards library. The author selects the Collect Data procedure from a list of stored procedures and drags it into the

new experiment block in flowchart 500, causing author interface module 140 to display experiment editor dialog 480 on output device 120. The author sets the remaining action parameters in the experiment editor dialog, including the Collect Data With Trigger Action's Library input parameter (set to the Standards Library object identified above) and XY Data Object output parameter (e.g., by entering "StandardsDataObject" in the Variables box in experiment editor dialog 480). Likewise, the author defines the remaining input parameters for the procedure's Inject Action (setting the Robot System Resource parameter to the RobotSystemResource object identified above and the Library parameter to Standards Library as discussed above) and the Collect Data With Trigger Action (setting the DataGatherer System Resource parameter to the DataGathererSystemResource object identified above). The author then links the newly-defined Collect Data for Standards block to Start Robot block 520, specifying that the former block is to be executed upon successful execution of the latter.

Next, the author adds an experiment block 530 that will contain the Detect Standards procedure created above. The author selects the Detect Standards procedure from a list of stored procedures and drags it into the new experiment block in flowchart 500, causing user interface module 140 to display experiment editor dialog 480 on output device 120. The author sets the remaining action parameters in the experiment editor dialog, including the Library ID input parameter (set to the Standards Library library object) and the Detect Single Peak Action's Peak Center output parameter (by entering a variable name, such as "PeakCenter"). Likewise, the author enters values for the Detect Single Peak Action's XY Data Object input parameter, by selecting the previously-defined StandardsDataObject object from the dropdown list of variables. The author then links the Detect Standards block to Collect Data for Standards block 525, specifying that the former block is to be executed upon successful execution of the latter.

Next, the author adds an experiment block 535 that will contain the Calibrate Standards procedure. The author selects the Calibrate Standards procedure from a list of stored procedures and drags it into the new experiment block in flowchart 500, causing user interface module 140 to display experiment editor dialog 480 on output device 120. The author selects a variable name for the Calibrate Action's Calibration Curve Object output parameter, which specifies a variable that will store the calibration curve generated by analysis of the standards (e.g., entering the variable name CalibrationCurve in experiment editor dialog 480). Likewise, the author enters values for the remaining input parameters, including the Calibrate Action's Library ID parameter (again, set to

Standards Library) and the Retention Time and MolWeight parameters, which set the standards' peak retention times and molecular weights, respectively (set, e.g., to the previously-defined PeakCenter and MolWeight variables). The author then links the Calibrate Standards block to the Detect Standards block 530, specifying that the former block is to be executed upon successful execution of the latter.

Similarly, to add the Collect Data for Samples procedure to the protocol, the author adds an experiment block 540, selects the Collect Data procedure from a list of stored procedures and drags it into the new experiment block in flowchart 500. The author then defines the remaining output and input parameters for the actions in the block, including the Collect Data With Trigger Action's Library ID input parameter (this time, set to Unknown Library) and DataGatherer System Resource input parameter (set to the DataGathererSystemResource object defined above), and XY Data Object output parameter (set to the SamplesDataObject object as described above), and the Inject Action's Library ID (Unknown Library) and Robot System Resource (set to the RobotSystemResource object defined above) input parameters. Once again, the author links the newly-defined Collect Data for Samples block to the previous Calibrate Standards block 535, specifying that the former block is to be executed upon successful execution of the latter.

Finally, the author adds an experiment block 545 to the protocol to contain the Calculate Results procedure, selects the Calculate Results procedure from the list of stored procedures and drags it into the new experiment block in flowchart 500. The author enters values for the Calculate Properties Action's Mw and Peak Center output parameters (the previously defined MolWeight and PeakCenter variables) and Library ID, Calibration Object and XY Data Object input parameters (the previously defined Unknown Library, CalibrationCurve and SamplesDataObject objects). The author links the newly-defined Calculate Results block to the previous Collect Data for Samples block 540, specifying that the former block is to be executed upon successful execution of the latter. The author then saves the completed protocol in database 170.

Referring to FIG. 6, a user (who may be the protocol's author or any other user of process control and data management system 130) instructs process control and data management system 130 to load a protocol by selecting the desired protocol from a list of stored protocols and entering a Load command (step 600). In response to an Execute command (step 610), protocol manager 150 executes the protocol block by block according to the sequence defined by the links created during protocol creation

(implemented, for example, as NextOnTrue and NextOnFalse properties of block component objects stored in database 170) (step 620). Within each block, protocol manager 150 executes each action in turn (step 630), generating as each action is executed a stream of process data that includes each of the action's input and output parameters (and any associated well count information) (step 640), as well as a return value indicating whether the action executed successfully (step 650).

If the block contains one or more actions that reference library objects (e.g., actions having "each well" or "all wells" well count properties as discussed above), protocol manager 150 retrieves the associated library object and either automatically iterates over the library defined by that object (for blocks containing actions having "each well" associated well counts) or executes each action once for each valid position in the library (for blocks containing "all wells" actions). In this way, protocols and procedures executed by process control and data management system 130 are not constrained to be executed on libraries having any specific geometry. An action whose methods perform activity on one or more library positions simply causes protocol manager 150 to retrieve a specified library object (designated, for example, by the action's input parameters as discussed above). All details of library geometry – including, for example, the geometry and location of the physical library substrate, the position of library wells on the substrate, and the identity of any invalid library positions (such as library positions flagged as having a problem by experimental instruments 110) – resides in the library object, not in the action, procedure or protocol. The output data stream also contains information about the library upon which the action's methods were performed, automatically associating the output data with the relevant library and library position(s).

If, for example, a physical library is prepared in a reactor 110 and reactor control software detects a problem in a particular library location, protocol manager 150 flags that position in the corresponding library object as invalid. When protocol manager 150 subsequently executes an analysis protocol to on the library, it automatically excludes the invalid library position when it executes a load library action, thereby allowing the analysis to skip this position automatically. Similarly, to apply a given protocol to a library having a different geometry, a user need only provide a corresponding library object, either during protocol definition or at run time, without modifying the protocol itself.

Library manipulation actions provide an additional measure of flexibility. A user selection library action permits a user to filter samples based on any desired criterion,

such as visual inspection (e.g., excluding highly colored samples) or to bifurcate a sample set for different analyses by protocol manager 150 (e.g., to specify one analysis for highly viscous samples and another for less viscous ones). An auto-selection library action automates this process, permitting, for example, a rapid primary screen of a library to determine a property crudely and then automatically selecting samples that meet a minimum threshold in the crude screen for a more time consuming secondary screening that yields more precise results (thereby increasing experimental throughput by subjecting only "interesting" samples to the full analysis).

When protocol manager 150 reaches the end of a block's action sequence (the NO branch of step 660), the block itself returns a true or false value based on the return values of its constituent actions; protocol manager 150 sends this value to the process data stream as well. If more blocks remain to be executed (the YES branch of step 670), protocol manager 150 then proceeds to execute the next block in the protocol. When protocol manager executes the last block in the sequence, the execution is complete (the NO branch of step 670).

Meanwhile, output parser 160 asynchronously monitors the process data stream, communicating all parameter values and return values to database services module 180, which stores the data in database 170. In one implementation, protocol manager 150 splits the data stream to populate two or more message queues that are monitored by output parser 160. In this implementation, input and output parameters are sent to an Action Output Queue, which includes all of the experimental results for the experiment, while return values are sent to an Execution Queue, which contains information concerning the execution status of each block and each action executed by protocol manager 150.

Execution of Calibrate Standards and Calculate Sample Peaks protocol 500 proceeds as follows. Protocol manager 150 first executes Load Library blocks 503 and 507, retrieving the Standards Library and Unknown Library objects from database 170 for subsequent use in the protocol.

Next, protocol manager 150 turns to Start Data Gatherer block 510. First, protocol manager 150 executes the block's Create DataGatherer Action, creating a Data Gatherer system resource object having a name and an acquisition frequency specified by the action's Data Gatherer Name and Acquisition Frequency input parameters and populating the action's Data Gatherer System Resource output parameter, identifying the data gatherer system resource object that is created. Upon successful execution, the

action returns a value of true; protocol manager 150 sends the return value to an Execution Queue, and the input and output parameter objects to an Action Output Queue, from which output parser 160 asynchronously retrieves the respective values and passes them to database services module 180 for storage in database 170. Protocol manager 150 then turns to the Initialize DataGatherer Action, initializing the data gatherer system resource object created by the previous action. Upon successful completion, the action returns a value of true, which, along with the action's DataGatherer System Resource input parameter, is passed to the appropriate message queues for retrieval and storage by output parser 160 and database services module 180. When Start Data Gatherer block 510 has been successfully executed, it, too, returns a value of true, which protocol manager 150 duly sends to the Execution Queue for retrieval and storage by output parser 160 and database services module 180.

After successful execution of Start Data Gatherer block 510, protocol manager 150 proceeds to the next block in the procedure, determined by the block's NextOnTrue property – here Start Robot block 520. Protocol manager 150 executes the Create Robot Action to create a robot system resource object named Local Robot as specified by the Robot Name input parameter, populating the Robot System Resource output parameter with the identity of the robot system resource that has been created. Upon successful creation, the action returns true and the action's parameters and return value are sent to the appropriate message queues. Protocol manager 150 then executes the Initialize Robot Action, causing a processor coupled to the material handling robot identified by the action's Robot System Resource parameter to retrieve and execute an initialization procedure identified by the action's Initialization Program Name input parameter. The action's return value is sent to the Execution Queue. When Start Robot block 520 has been successfully executed, its return value is sent to the Execution Queue and eventually stored in database 170.

Next, protocol manager 150 executes Collect Data for Standards block 525. Protocol manager 150 first executes the Inject Action, causing a processor coupled to the robot corresponding to the Robot System Resource parameter to retrieve and execute to the Inject Standards program specified in the action's Injection Program Name parameter, iterating through the wells in Standards Library and injecting the corresponding standard materials into the injection port specified by the action's Injection Volume and Injection port parameters. After sending the Inject Action's parameters and return value to the message queues, protocol manager 150 executes

the Collect Data With Trigger Action, causing the evaporative light scattering detector specified by the action's Data Gatherer System Resource parameter to begin waiting for the trigger voltage specified by the action's Trigger Voltage parameter. If a voltage pulse is not detected within the timeout period specified by the action's Trigger Timeout parameter, the action immediately returns false. If a voltage pulse is detected within the timeout window, the action immediately returns true and the data is collected asynchronously for the amount of time specified by the action's Total Acquisition Time parameter, including any specified post-trigger delay (set in the Post-Trigger Delay parameter). The collected data is sent to the appropriate message queue as the StandardsDataObject variable specified by the action's XY Data Object output parameter (as are the action's input parameters and return value). The appropriate parameter and return values are sent to the appropriate message queues for storage.

Next, protocol manager 150 executes Detect Standards block 530 to detect the peaks in the Standards Library and executes Calibrate Standards block 535 to generate a calibration curve from the peak data. Protocol manager 150 executes the Detect Single Peak Action, using the values in the StandardsDataObject variable to identify peaks from the data collected from Standards Library. Protocol manager 150 sends the resulting peak information (in the form of the Peak Height, Peak Area, and Peak Width output parameters) to the Action Output Queue, and sends the appropriate return values to the Execution Queue. Protocol manager 150 then executes the Calibrate Action applying the specified transformation (set by the Transform Type input parameter) to the molecular weight and retention time data generated in the preceding block to generate a calibration curve of the type specified in the action's Curve Type input parameter. Protocol manager 150 assigns the resulting calibration curve to the action's CalibrationCurve object specified by the Calibration Curve Object output parameter, which is sent to the Action Output Queue for storage, as are the action's input parameters and the return values for the Calibrate Action and Calibrate Standards block 535.

When the detector has been successfully calibrated, protocol manager 150 executes Collect Data for Samples block 540. Protocol manager 150 first executes the Inject Action, causing the processor coupled to the material handling robot specified by the Robot System Resource parameter to retrieve and execute the Inject Samples program specified in the Injection Program Name parameter, iterating over the library positions identified by the Unknown Library variable and injecting amounts of the corresponding

polymer samples into the injection port specified by the action's Injection Volume and Injection port parameters, respectively. When this action has been executed (and the corresponding parameters and return values sent to the appropriate message queues), protocol manager 150 executes the Collect Data With Trigger Action, causing the evaporative light scattering detector specified by the action's Data Gatherer System Resource parameter to begin waiting for the trigger voltage specified by the action's Trigger Voltage parameter. If a voltage pulse is not detected within the timeout period specified by the action's Trigger Timeout parameter, the action immediately returns false. If a voltage pulse is detected within the timeout window, the action immediately returns true and the data is collected asynchronously for the amount of time specified by the action's Total Acquisition Time parameter, including any specified post-trigger delay (set in the Post-Trigger Delay parameter). The return value, input parameters and collected sample data, in the form of the SamplesDataObject object specified by the action's XY Data Object output parameter, are sent to the appropriate message queues.

Finally, protocol manager 150 executes Calculate Results block 545 to calculate molecular weight and peak area values for the polymer samples. Protocol manager 150 executes the Calculate Properties Action to detect and integrate the unknown peaks using known techniques. When the calculation is complete, process control manager 150 sends the input parameters and the molecular weight and peak area output parameters to the Action Output Queue and the action and block return values to the Execution Queue for storage in database 170.

When a protocol is executed, process control and data management system 130 creates an experiment record in database 170 that records, in order, every action that occurred during the experiment. Through input/output devices 120 and user interface module 140, a user may access the experiment record – either during execution, to monitor the experiment's progress, or after execution is complete, to reconstruct or play back the experiment for further analysis. In one implementation, user interface 140 is configured to access Execution and Action Output message queues through database services module 180 and display the queues (and their constituent objects) on output device 120. In addition, user interface 140 is configured to access pre-defined display objects stored in database 170, which objects are operable to create a dialog on output device 120 (such as an ATL dialog object hosting composite Active X controls written in Visual Basic) and display the output from one or more specific objects in an experiment protocol. Thus, for example for the Calibrate Standards and Calculate Sample Peaks

protocol example discussed above, user interface 140 can be configured to access display objects that support the Collect Data With Trigger or Calculate Properties Actions, which, when output data from the supported actions is detected, cause user interface 140 to create an appropriate dialog and post the actions' XY Data Object or molecular weight and peak area output parameters for viewing on output device 120.

In addition to executing protocols generating experimental data from experimental instruments 110, process control and data management system 130 can execute protocols that retrieve experimental data from database 170 for analysis. For example, after executing Calibrate Standards and Calculate Sample Peaks protocol 500 as discussed above, process control and data management system 130 can execute a protocol that extracts the raw peak data (for example, in units of $\mu V/t$) by retrieving the SamplesDataObject object from database 170 and reanalyzes the data – for example, by applying a different calibration curve to calculate molecular weights from the raw data – just as if the data had been received directly from the experimental instrument.

The invention can be implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations of them. Apparatus of the invention can be implemented in a computer program product tangibly embodied in a machine-readable storage device for execution by a programmable processor; and method steps of the invention can be performed by a programmable processor executing a program of instructions to perform functions of the invention by operating on input data and generating output. The invention can be implemented advantageously in one or more computer programs that are executable on a programmable system including at least one programmable processor coupled to receive data and instructions from, and to transmit data and instructions to, a data storage system, at least one input device, and at least one output device. Each computer program can be implemented in a high-level procedural or object-oriented programming language, or in assembly or machine language if desired; and in any case, the language can be a compiled or interpreted language. The protocols, procedures, blocks, actions and other objects discussed above can be implemented as component objects implementing an appropriate interface in a component software architecture such as Microsoft Corporation's Component Object Model (COM) or Distributed Component Object Model (DCOM) standards, or the Object Management Group's Common Object Request Broker Architecture (CORBA) standard. Suitable processors include, by way of example, both general and special purpose microprocessors. Generally, a processor will receive instructions and data from a

read-only memory and/or a random access memory. Generally, a computer will include one or more mass storage devices for storing data files; such devices include magnetic disks, such as internal hard disks and removable disks; magneto-optical disks; and optical disks. Storage devices suitable for tangibly embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, such as EPROM, EEPROM, and flash memory devices; magnetic disks such as internal hard disks and removable disks; magneto-optical disks; and CD-ROM disks. Any of the foregoing can be supplemented by, or incorporated in, ASICs (application-specific integrated circuits).

To provide for interaction with a user, the invention can be implemented on a computer system having a display device such as a monitor or LCD screen for displaying information to the user and a keyboard and a pointing device such as a mouse or a trackball by which the user can provide input to the computer system. The computer system can be programmed to provide a graphical user interface through which computer programs interact with users.

A number of implementations of the invention have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the invention. For example, while the system has been described as being implemented in data objects stored in a database, those skilled in the art would recognize that alternative data structures and storage formats could be used. Accordingly, other embodiments are within the scope of the following claims.

WHAT IS CLAIMED IS:

1. A computer-implemented process control and data management system for processing a set of experiments on a combinatorial library of materials, the process control and data management system comprising:

a memory storing one or more experiment protocols, each experiment protocol defining a program of actions to be carried out during a set of experiments, the program of actions including a first action linking the experiment protocol to a library description stored in the memory, the library description being data including data identifying a plurality of library positions corresponding to a plurality of members of the combinatorial library; and

a protocol manager operable to receive and execute an experiment protocol from the memory, such that upon execution of the first action, the protocol manager is operable to perform a method specified by the first action using the library description as an input.

2. The process control and data management system of claim 1, wherein:

the first action is operable to cause the protocol manager to perform a specified method upon a plurality of the library positions identified by the library description.

3. The process control and data management system of claim 1, wherein:

the first action is operable to cause the protocol manager to perform a specified method sequentially upon a plurality of the library positions identified by the library description.

4. The process control and data management system of claim 1, wherein:

the protocol manager is operable to receive an input identifying a combinatorial library member as invalid and modify the library description to remove the corresponding library position from the library description.

5. The process control and data management system of claim 1, wherein:

the experiment protocol includes a library creator action operable to cause the protocol manager to generate a library description corresponding to a specified library.

6. The process control and data management system of claim 1, wherein:

the experiment protocol includes a library loader action operable to cause the protocol manager to retrieve a library description from the memory.

7. The process control and data management system of claim 1, wherein:

the experiment protocol includes a library manipulator action operable to generate at least one output library description from at least one input library description.

8. The process control and data management system of claim 7, wherein:

the library manipulator action is operable to cause the protocol manager to retrieve the input library description from the memory; display a graphical representation of the input library description on a user interface; and receive an input from a user defining one or more output library descriptions based on the displayed representation.

9. The process control and data management system of claim 7, wherein:

the library manipulator action is operable to cause the protocol manager to retrieve the input library description from the memory and apply a selection criterion to define one or more output library descriptions based on the input library description.

10. The process control and data management system of claim 1, wherein:

the protocol manager is operable to generate during execution of an experiment protocol a stream of process data including output data received from one or more experimental instruments coupled to the protocol manager.

11. The process control and data management system of claim 10, further comprising:

an output parser operable to retrieve the stream of process data and store the process data in the memory.

12. The process control and data management system of claim 1, wherein:

each experiment protocol includes a plurality of blocks, each block including at least one action operable upon execution by the protocol manager to receive input data and generate output data, each block also including instructions for sequentially passing input data to and receiving output data from the actions in the block.

13. The process control and data management system of claim 12, wherein:

the protocol manager is operable to execute a first block in the protocol, such that as each action in the first block is executed the protocol manager adds the corresponding output data to the process data stream.

14. The process control and data management system of claim 12, wherein:

the memory stores process control objects including one or more display objects operable to cause the protocol manager to display the output data generated by an action in a protocol; and

the protocol manager is operable to execute a display object such that the output data from an action in an executing protocol is displayed on an output device in a format specified by the display object.

15. The process control and data management system of claim 13, wherein:

the protocol manager is operable to execute upon completion of execution of the first block a second block identified from the plurality of blocks in the protocol by a flow control instruction generated upon completion of execution of the first block.

16. The process control and data management system of claim 15, wherein:

the protocol manager is operable to generate execution status data upon execution of each action and each block and add the execution status data to the process data stream, the execution status data designating an execution status for the action or block.

17. The process control and data management system of claim 16, wherein:

the protocol manager is operable upon execution of the first block to use the execution status data for the block as the flow control instruction identifying the second block to be executed from the plurality of blocks in the protocol.

18. The process control and data management system of claim 14, wherein:

the memory includes a library of actions, each action being defined by action methods and action data; the system further comprising:

a user interface module operable in response to a user input to cause the protocol manager to create one or more blocks and to include in each block one or more actions selected from the library of actions and to define a flow of control among the one or more blocks.

19. The process control and data management system of claim 18, wherein:

the library of actions includes one or more detector objects operable upon execution to retrieve experimental data from a detector and to provide the experimental data as output data to the protocol manager.

20. The process control and data management system of claim 19, wherein:

the library of actions includes one or more result objects operable upon execution to receive experimental data from a detector object, calculate one or more properties from the experimental data, and provide the calculated properties as output data to the protocol manager.

21. The process control and data management system of claim 12, wherein:

the protocol manager is operable to execute an experiment protocol to carry out an experiment on a combinatorial library of materials, the protocol including a block operable to test the output data stream, define a new library description that corresponds to a subset of the combinatorial library of materials, and to cause at least a subset of the blocks of the experiment protocol to operate on the subset of the combinatorial library of materials corresponding to the new library description.

22. A computer-implemented method for processing a set of experiments on a combinatorial library of materials, the method comprising:

providing an experiment protocol defining a program of actions to be carried out during a set of experiments, the protocol including at least a first action being linked to a library description of a combinatorial library, the library description being data including data identifying a plurality of library positions corresponding to a plurality of members of the combinatorial library; and

executing the program of actions, wherein execution of the first action comprises performing a method specified by the first action using the library description as an input.

23. The method of claim 22, wherein the first action is linked to the library description at run time.

24. The method of claim 22, wherein:

the specified method is performed upon all of the library positions identified by the library description.

25. The method of claim 22, wherein:

the specified method is performed sequentially upon each of the library positions identified by the library description.

26. The method of claim 22, further comprising:

receiving an input identifying a combinatorial library member as invalid; and

modifying the library description to remove the corresponding library position from the library description.

27. The method of claim 22, wherein:

the experiment protocol includes a library creator action operable to cause the protocol manager to generate a library description corresponding to a specified library and store the library description in a memory.

28. The method of claim 22, wherein:

the experiment protocol includes a library loader action operable to retrieve a library description from a memory.

29. The method of claim 22, wherein:

the experiment protocol includes a library manipulator action operable to generate at least one output library description from at least one input library description.

30. The method of claim 29, further comprising:

upon execution of the library manipulator action, retrieving the input library description from a memory; displaying a graphical representation of the input library description on a user interface; and receiving an input from a user defining one or more output library descriptions based on the displayed representation.

31. The method of claim 29, further comprising:

upon execution of the library manipulator action, retrieving the input library description from a memory and applying a selection criterion to define one or more output library descriptions based on the input library description.

32. The method of claim 22, wherein:

the protocol includes a plurality of blocks, each block specifying one or more actions and instructions for sequentially passing input data to and receiving output data from each action in the block.

33. The method of claim 32, executing the program of actions further comprising:

as each action in a first block is executed, adding the corresponding output data to a process data stream; and

upon completion of execution of the first block, executing a second block in the protocol according to a flow control instruction determined upon completion of execution of the first block.

34. The method of claim 33, further comprising:

asynchronously retrieving the output data from the process data stream and storing the output data in a memory.

35. The method of claim 32, further comprising:

providing a display object operable upon execution to display the output data associated with an action in the protocol; and

displaying the output data on an output device in a format specified by the display object.

36. The method of claim 33, further comprising:

upon execution of each block in the protocol and each action in each block, generating execution status data identifying a completion status for the corresponding block or action;

adding the execution status data to the process data stream; and

asynchronously retrieving the execution status data from the process data stream .

37. The method of claim 33, further comprising:

providing a library of pre-defined actions; and wherein:

providing the experiment protocol includes receiving user input defining an arrangement of blocks in the protocol and a selection and arrangement of actions in each block from the library of actions.

38. A computer-implemented method for processing a set of experimental data derived from a combinatorial library of materials, the method comprising:

providing an experiment protocol including a program of actions for processing a set of experimental data, the experiment protocol including at least a first action being linked to a library description of a combinatorial library, the library description being data including data identifying a plurality of library positions corresponding to a plurality of members of the combinatorial library, and a second action specifying a data processing method;

in response to execution of the first action, retrieving a set of experimental data derived from the combinatorial library; and

in response to execution of the second action, performing the specified data processing method on the set of experimental data.

39. The method of claim 38, wherein:

the set of experimental data is retrieved from an experimental instrument.

40. The method of claim 38, wherein:

the set of experimental data is retrieved from a memory.

41. A computer program product, tangibly stored on a computer-readable medium, for processing a set of experiments on a combinatorial library of materials, the computer program comprising instructions operable to cause a programmable processor to:

provide an experiment protocol defining a program of actions to be carried out during a set of experiments, the protocol including at least a first action being linked to a library description of a combinatorial library, the library description being data including data identifying a plurality of library positions corresponding to a plurality of members of the combinatorial library; and

execute the program of actions, wherein execution of the first action comprises performing a method specified by the first action using the library description as an input.

42. The computer program of claim 41, wherein the first action is linked to the library description at run time.

43. The computer program of claim 41, wherein:

the instructions operable to cause a programmable processor to execute the action linked to the library description include instructions operable to cause a programmable

processor to perform a specified method upon all of the library positions identified by the library description.

44. The computer program of claim 41, wherein:

the instructions operable to cause a programmable processor to execute the action linked to the library description include instructions operable to cause a programmable processor to perform a specified method sequentially upon each of the library positions identified by the library description.

45. The computer program of claim 41, further comprising instructions operable to cause a programmable processor to:

receive an input identifying a combinatorial library member as invalid; and

modify the library description to remove the corresponding library position from the library description.

46. The computer program of claim 41, wherein:

the experiment protocol includes a library creator action operable to cause the protocol manager to generate a library description corresponding to a specified library and store the library description in a memory.

47. The computer program of claim 41, wherein:

the experiment protocol includes a library loader action operable to retrieve a library description from a memory.

48. The computer program of claim 41, wherein:

the experiment protocol includes a library manipulator action operable to generate at least one output library description from at least one input library description.

49. The computer program of claim 48, further comprising instructions operable to cause a programmable processor to:

retrieve the input library description from a memory upon execution of the library manipulator action; display a graphical representation of the input library description on a user interface; and receive an input from a user defining one or more output library descriptions based on the displayed representation.

50. The computer program of claim 48, further comprising instructions operable to cause a programmable processor to:

retrieve the input library description from a memory upon execution of the library manipulator action and apply a selection criterion to define one or more output library descriptions based on the input library description.

51. The computer program of claim 41, wherein:

the protocol includes a plurality of blocks, each block specifying one or more actions and instructions for sequentially passing input data to and receiving output data from each action in the block.

52. The computer program of claim 52, the instructions operable to cause a programmable processor to execute the program of actions further comprising instructions operable to cause a programmable processor to:

add the corresponding output data to a process data stream as each action in a first block is executed; and

execute a second block in the protocol according to a flow control instruction determined upon completion of execution of the first block.

53. The computer program of claim 52, further comprising instructions operable to cause a programmable processor to:

asynchronously retrieve the output data from the process data stream; and

store the output data in a memory.

54. The computer program of claim 51, further comprising instructions operable to cause a programmable processor to:

provide a display object operable upon execution to display the output data associated with an action in the protocol; and

display the output data on an output device in a format specified by the display object.

55. The computer program of claim 52, further comprising instructions operable to cause a programmable processor to:

upon execution of each block in the protocol and each action in each block, generate execution status data identifying a completion status for the corresponding block or action;

add the execution status data to the process data stream; and

asynchronously retrieve the execution status data from the process data stream .

56. The computer program of claim 52, further comprising instructions operable to cause a programmable processor to:

provide a library of pre-defined actions; and wherein:

the instructions operable to provide the experiment protocol include instructions operable to cause a programmable processor to receive user input defining an arrangement of blocks in the protocol and a selection and arrangement of actions in each block from the library of actions.

57. A computer program product, tangibly stored on a computer-readable medium, for processing a set of experimental data derived from a combinatorial library of materials, the computer program comprising instructions operable to cause a programmable processor to:

provide an experiment protocol including a program of actions for processing a set of experimental data, the experiment protocol including at least a first action being linked to a library description of a combinatorial library, the library description being data including data identifying a plurality of library positions corresponding to a plurality of members of the combinatorial library, and a second action specifying a data processing method;

retrieve a set of experimental data derived from the combinatorial library in response to execution of the first action; and

perform the specified data processing method on the set of experimental data in response to execution of the second action.

58. The computer program of claim 57, wherein:

the set of experimental data is retrieved from an experimental instrument.

59. The computer program of claim 57, wherein:

the set of experimental data is retrieved from a memory.

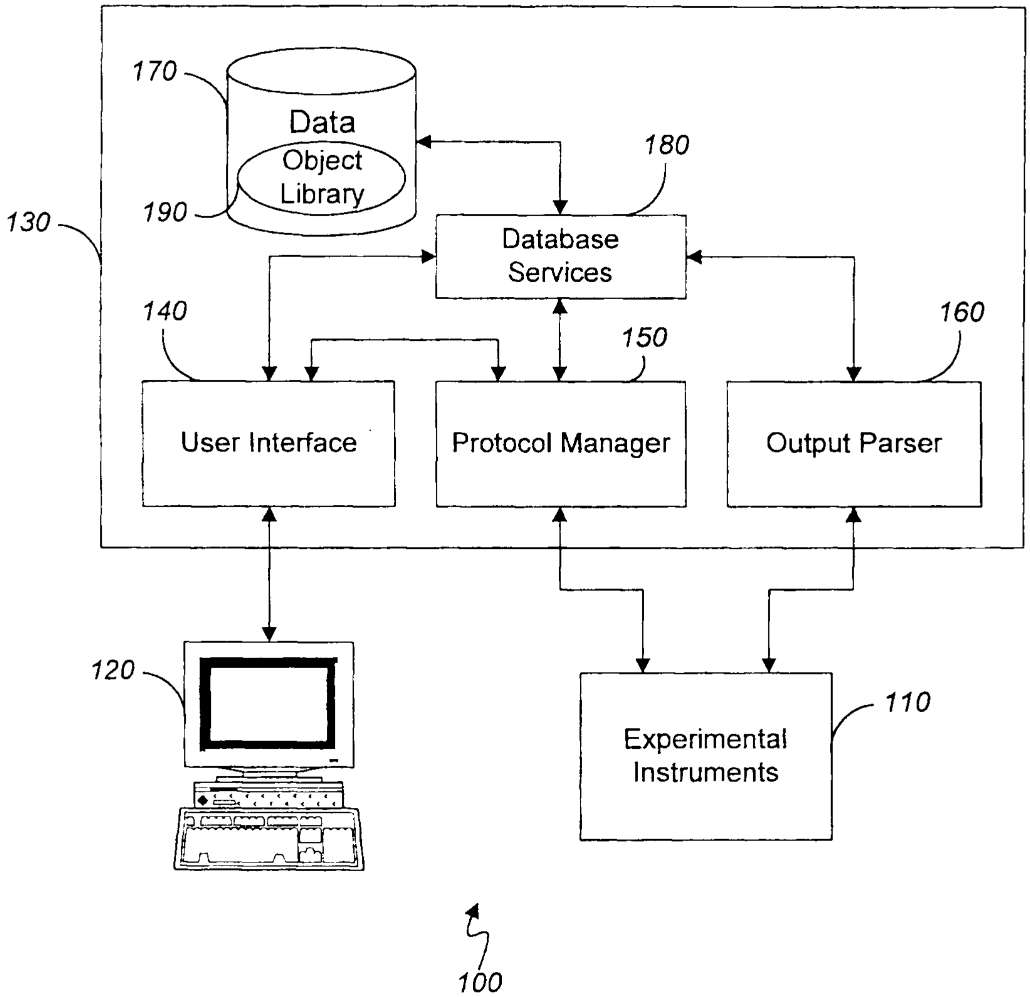

170
Data
Object
Library
190
180
130
Database
Services
140
150
160
User Interface
Protocol Manager
Output Parser
120
Experimental
Instruments
110
100

FIG. 1

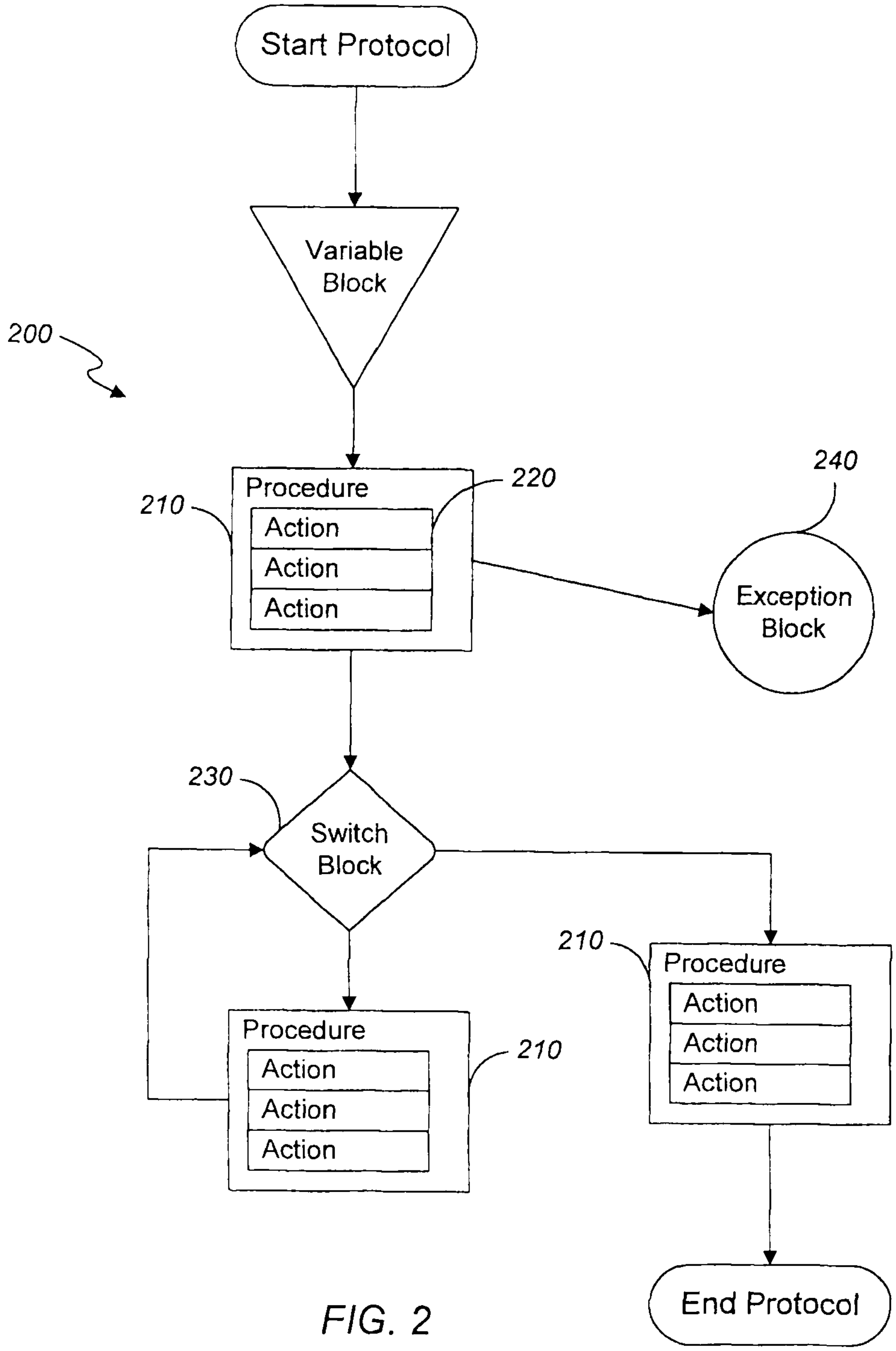

Start Protocol
Variable Block
200
Procedure
Action
Action
Action
210
220
240
Exception Block
230
Switch Block
210
Procedure
Action
Action
Action
Procedure
Action
Action
Action
210
End Protocol

FIG. 2

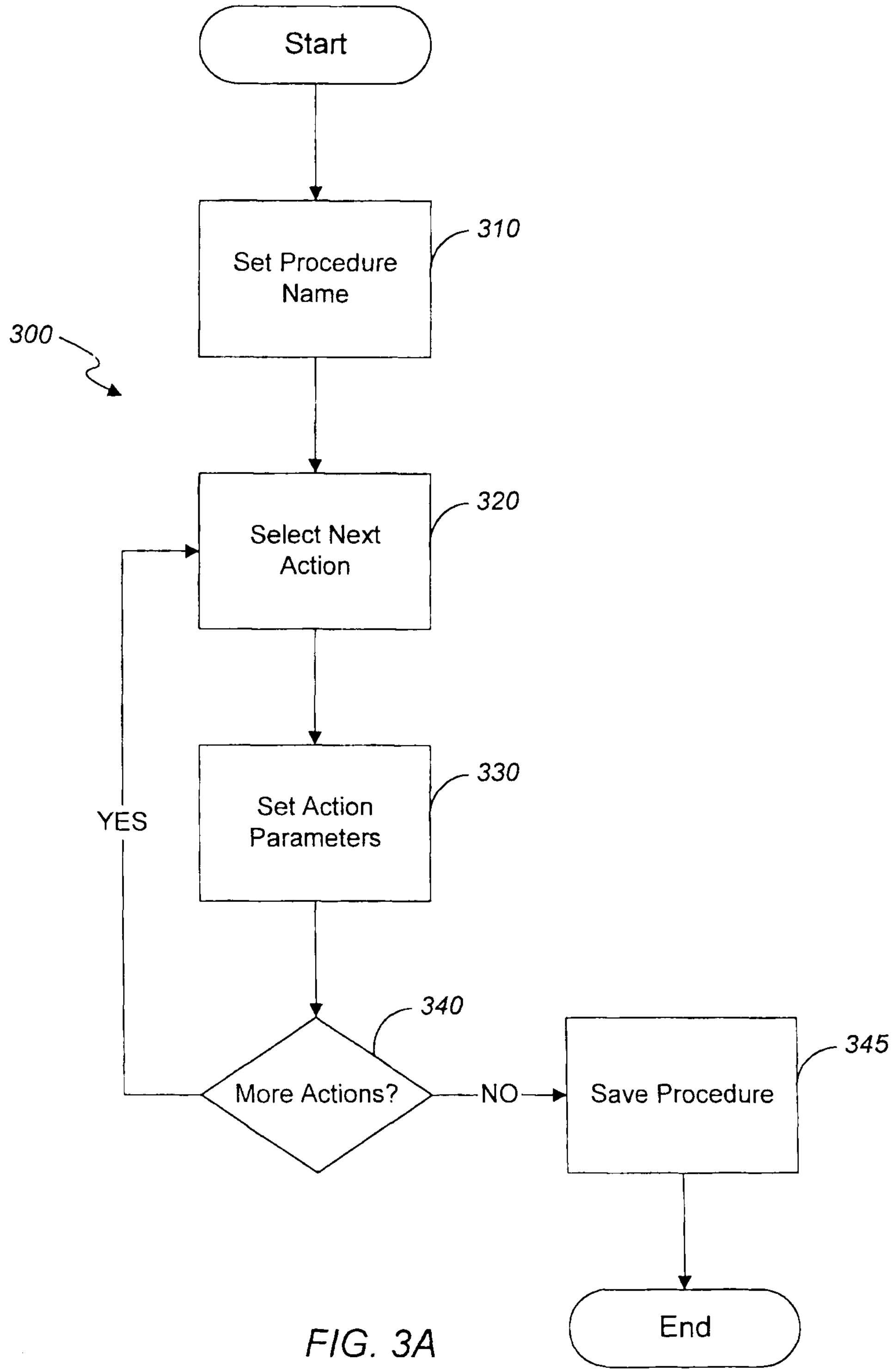

Start
Set Procedure Name
310
300
Select Next Action
320
Set Action Parameters
330
YES
More Actions?
340
NO
Save Procedure
345
End

FIG. 3A

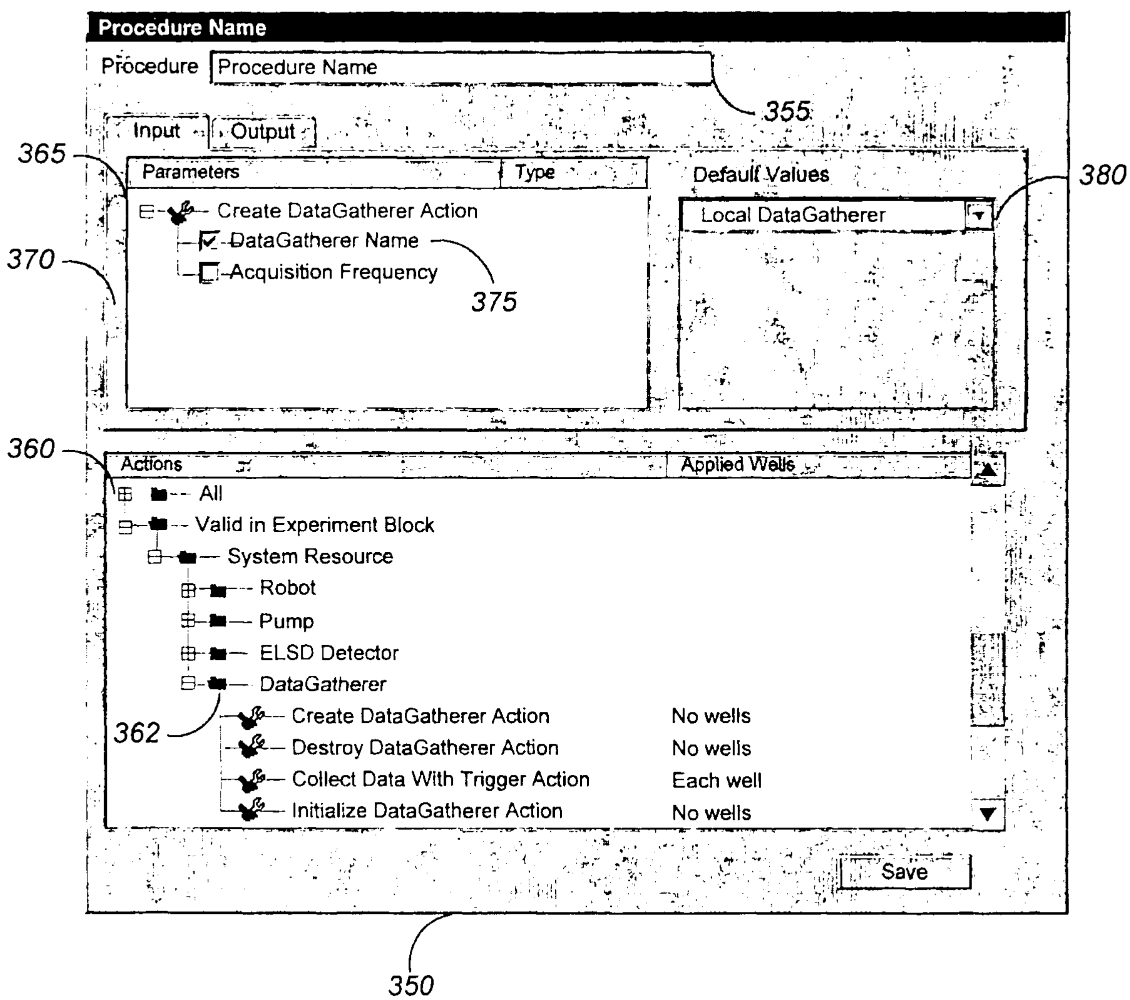
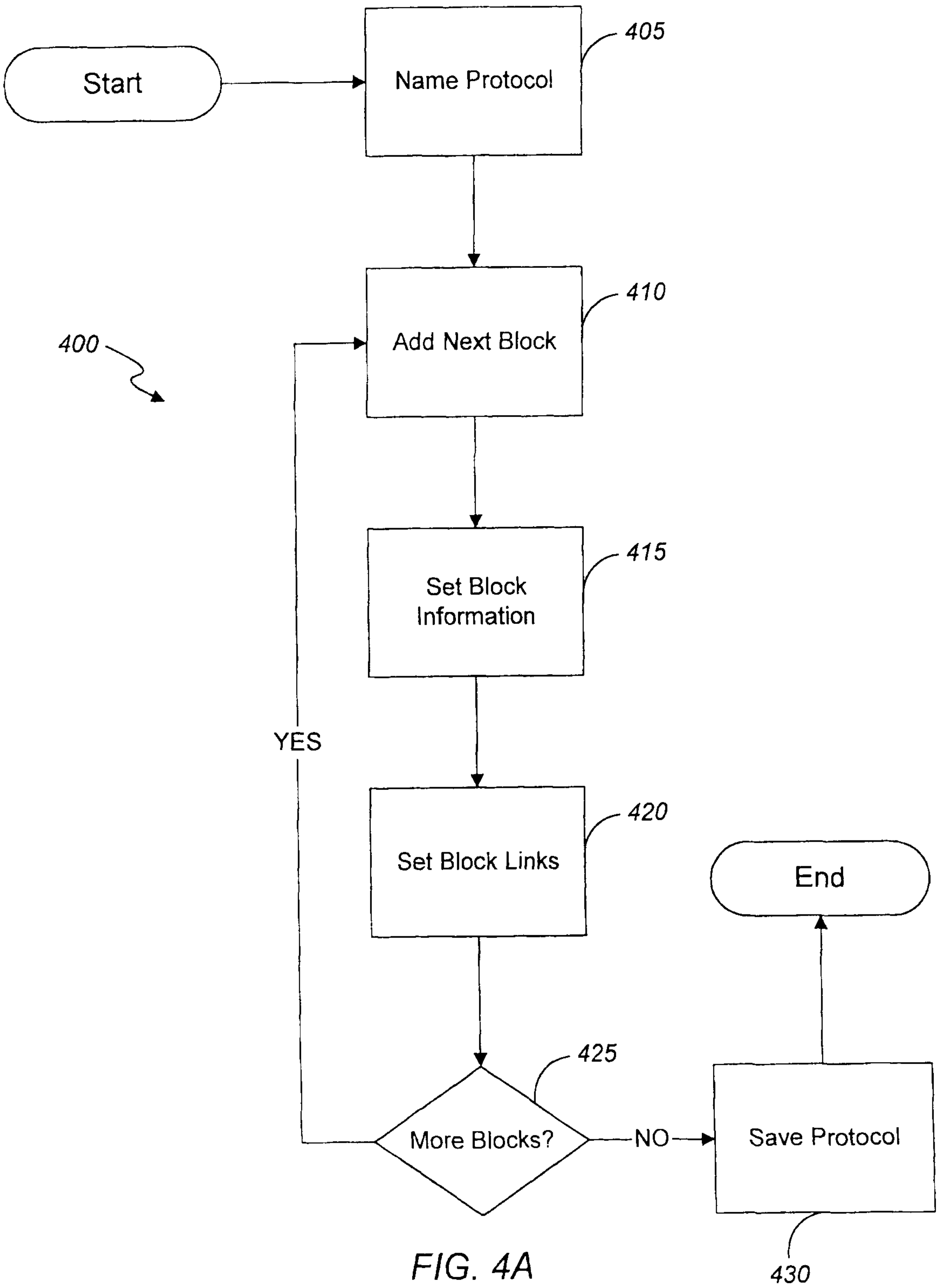
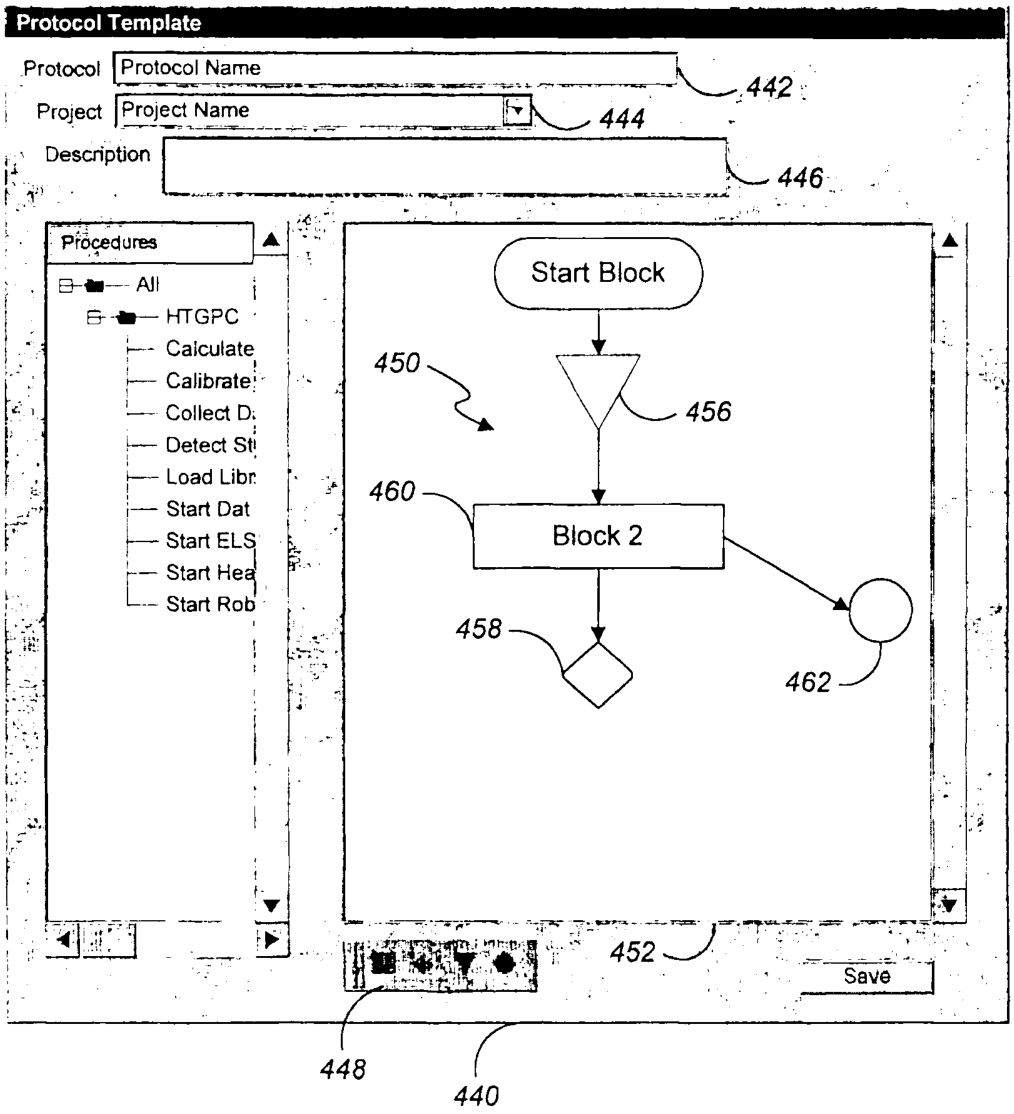

Procedure Name
Procedure
Procedure Name
355
Input
Output
365
Parameters
Type
Default Values
380
Create DataGatherer Action
DataGatherer Name
Acquisition Frequency
Local DataGatherer
370
375
360
Actions
Applied Wells
All
Valid in Experiment Block
System Resource
Robot
Pump
ELSD Detector
DataGatherer
362
Create DataGatherer Action
No wells
Destroy DataGatherer Action
No wells
Collect Data With Trigger Action
Each well
Initialize DataGatherer Action
No wells
Save
350

FIG. 3B



Start
Name Protocol
405
Add Next Block
410
400
Set Block Information
415
YES
Set Block Links
420
End
425
More Blocks?
NO
Save Protocol
430

FIG. 4A



Protocol Template
Protocol
Protocol Name
442
Project
Project Name
444
Description
446
Procedures
All
HTGPC
Calculate
Calibrate
Collect D
Detect St
Load Libr
Start Dat
Start ELS
Start Hea
Start Rob
Start Block
450
456
460
Block 2
458
462
452
Save
448
440

FIG. 4B

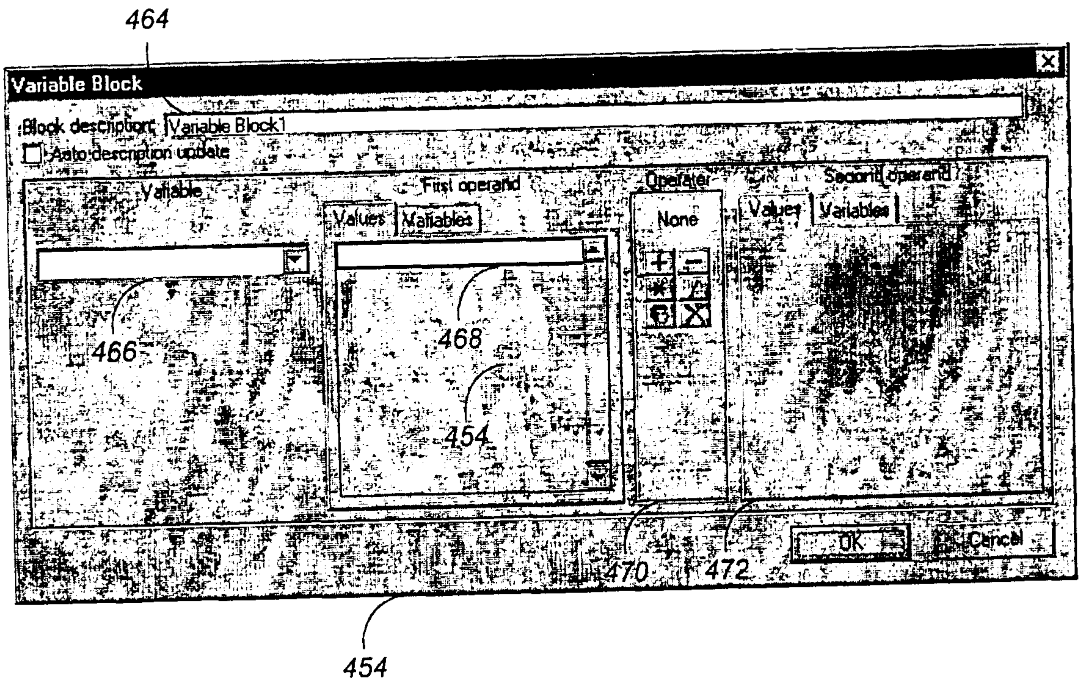

464
Variable Block
Block description: Variable Block1
Auto description update
Variable
First operand
Values
Variables
Operator
None
Second operand
Values
Variables
466
468
454
470
472
OK
Cancel
454

FIG. 4C

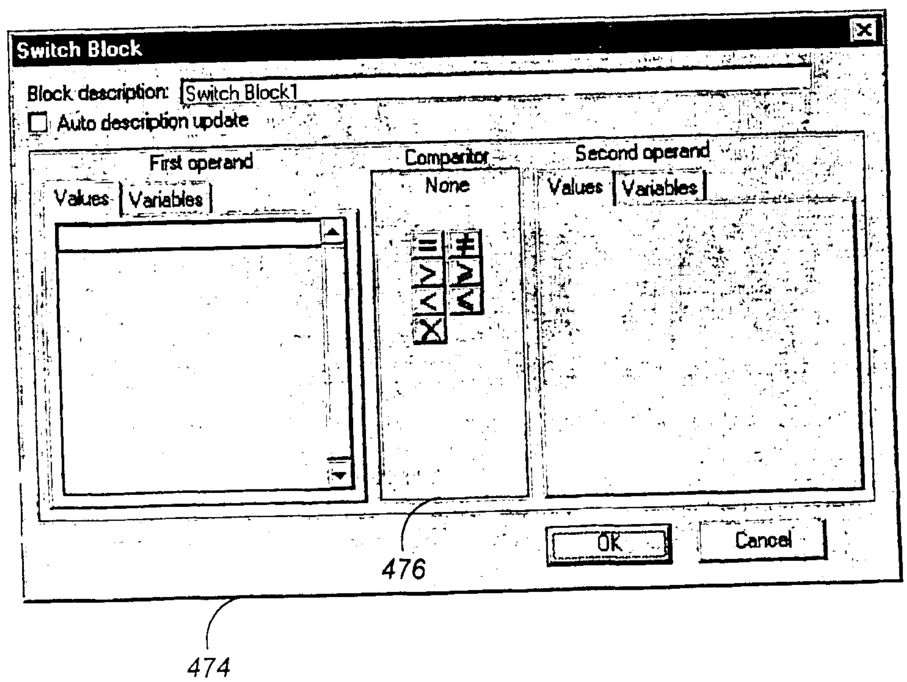

Switch Block
Block description: Switch Block1
Auto description update
First operand
Values
Variables
Comparitor
None
Second operand
Values
Variables
476
OK
Cancel
474

FIG. 4D

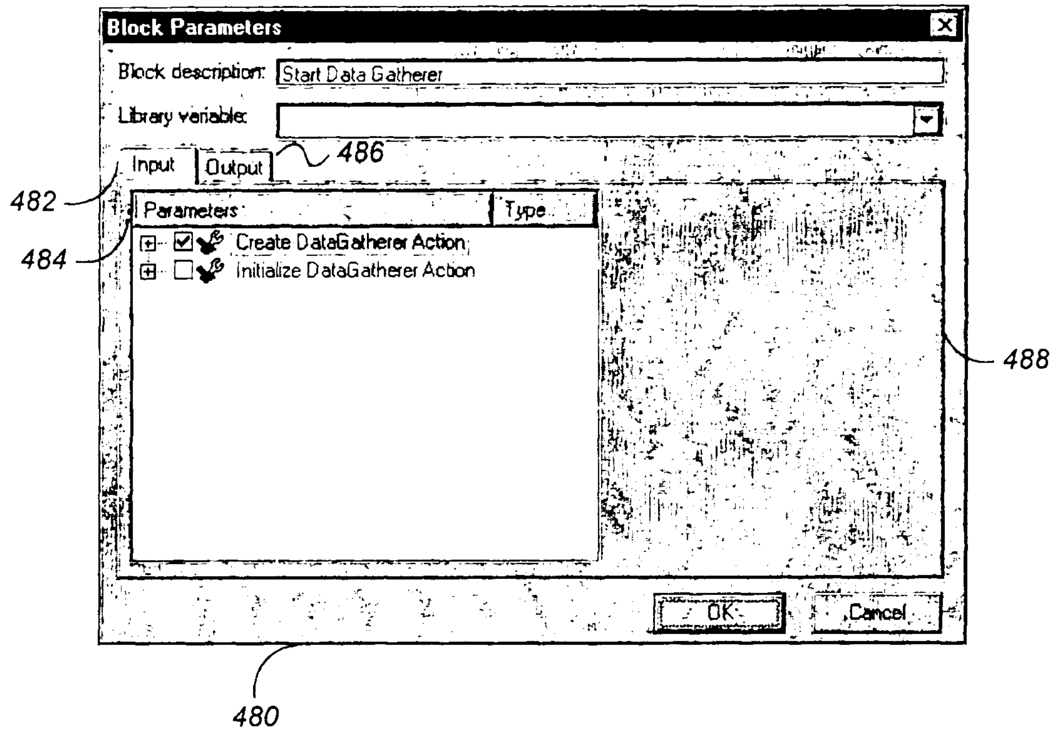

Block Parameters
Block description: Start Data Gatherer
Library variable:
Input
Output
486
482
484
Parameters
Type
Create DataGatherer Action
Initialize DataGatherer Action
488
OK
Cancel
480

FIG. 4E

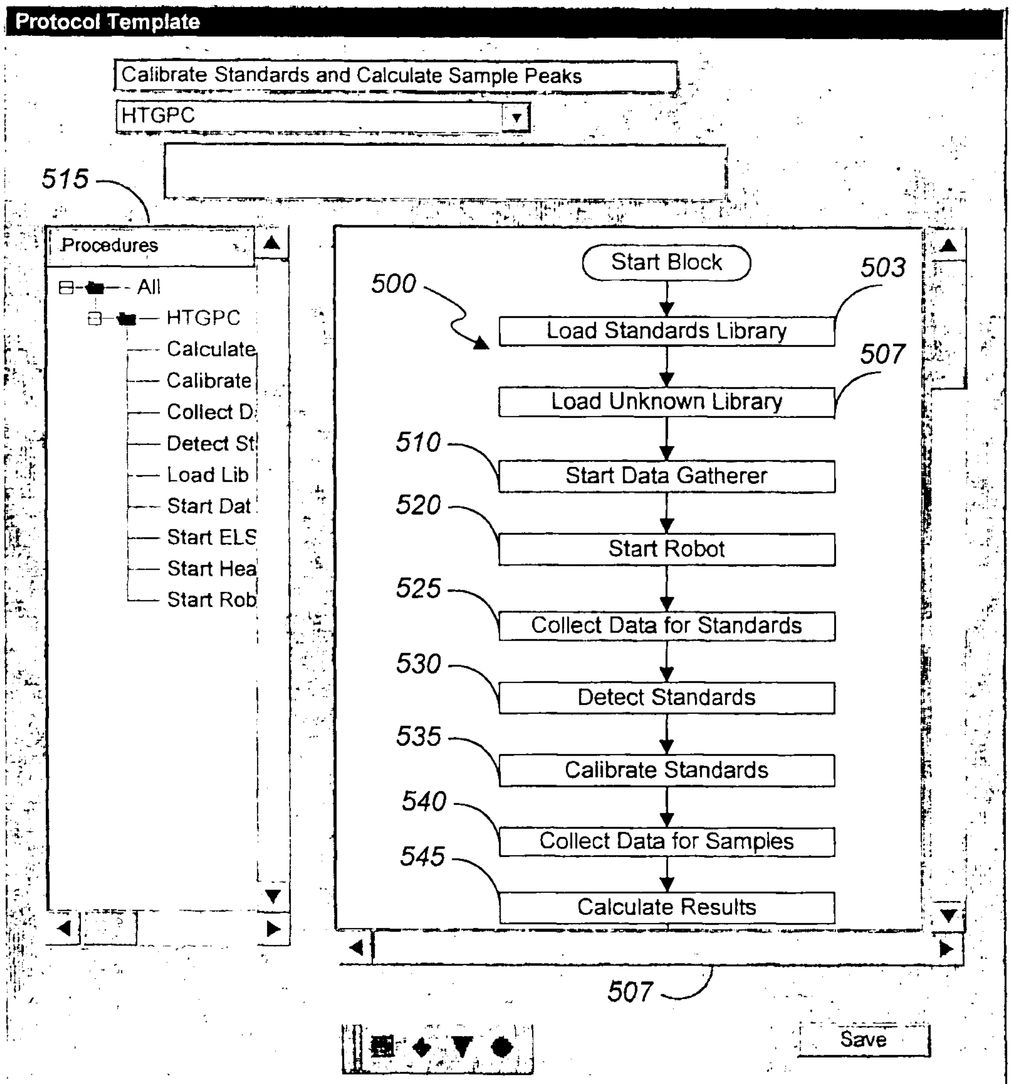

Protocol Template
Calibrate Standards and Calculate Sample Peaks
HTGPC
515
Procedures
All
HTGPC
Calculate
Calibrate
Collect D
Detect St
Load Lib
Start Dat
Start ELS
Start Hea
Start Rob
500
Start Block
503
Load Standards Library
507
Load Unknown Library
510
Start Data Gatherer
520
Start Robot
525
Collect Data for Standards
530
Detect Standards
535
Calibrate Standards
540
Collect Data for Samples
545
Calculate Results
507
Save

FIG. 5

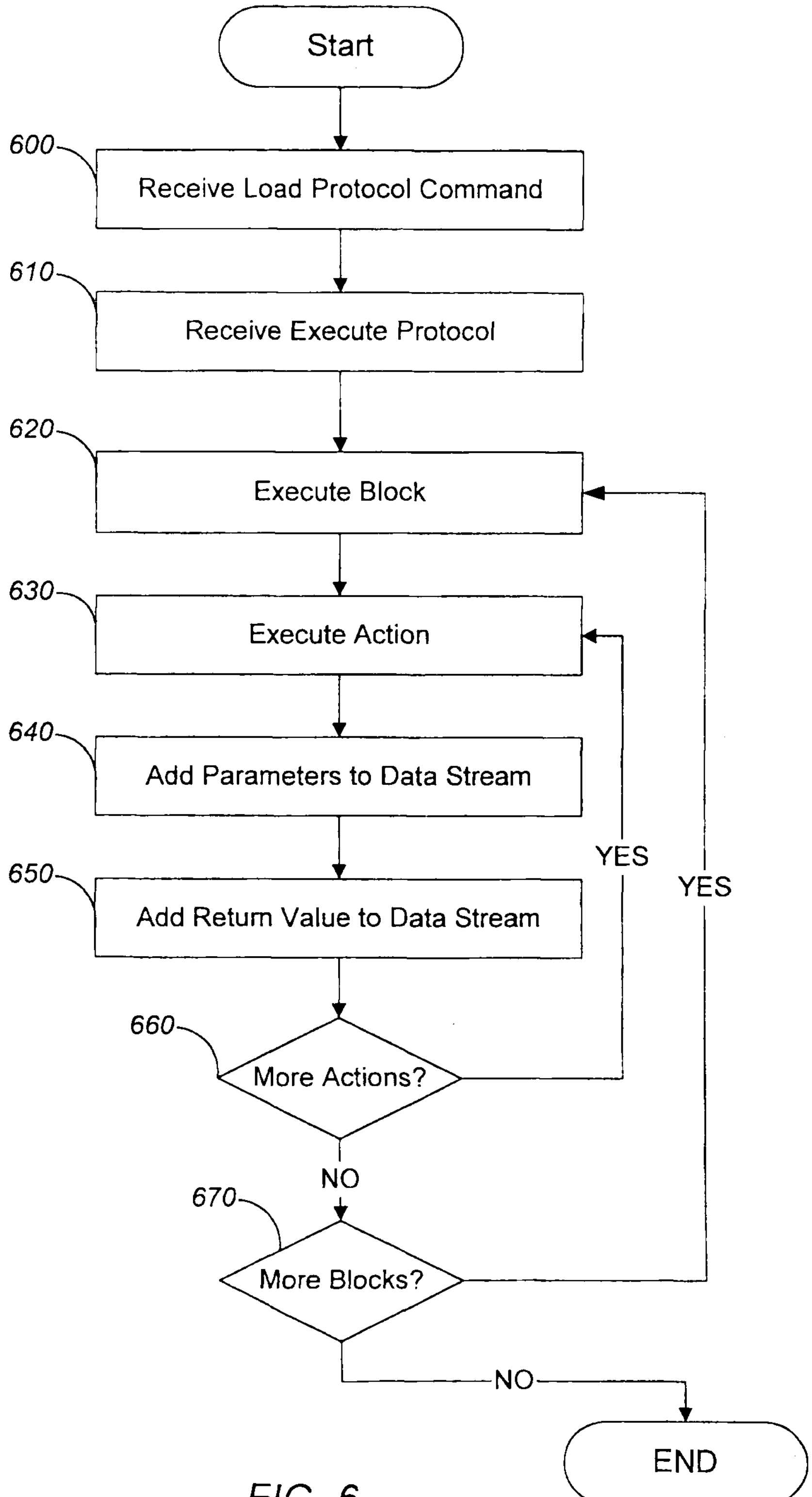

Start
600
Receive Load Protocol Command
610
Receive Execute Protocol
620
Execute Block
630
Execute Action
640
Add Parameters to Data Stream
650
Add Return Value to Data Stream
660
More Actions?
YES
NO
670
More Blocks?
YES
NO
END

FIG. 6